2003

Multispectral texture synthesis

J. Alexander Tyrrell

Multispectral Texture Synthesis

by

J. Alexander Tyrrell

Previous Degree
Sc.B Brown University, 1996

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Department of Computer Science
Rochester Institute of Technology

January 8, 2003

Signature of the Author _____

Accepted by B. Hirsh 1/29/03
Coordinator, M.S. Degree Program Date

DEPARTMENT OF COMPUTER SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of J. Alexander Tyrrell
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
M.S. degree in Computer Science

Peter G. Anderson, Thesis Advisor

John R. Schott

Harvey Rhody

Date January 29, 2003

THESIS RELEASE PERMISSION
ROCHESTER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Title of Thesis:

Multispectral Texture Synthesis

I, J. Alexander Tyrrell, hereby grant permission to Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature _____ Date January 29, 2003

Multispectral Texture Synthesis

J. Alexander Tyrrell

28 January, 2003

Contents

List of Figures

List of Tables

1 Abstract

Synthesizing texture involves the ordering of pixels in a 2D arrangement so as to display certain known spatial correlations, generally as described by a sample texture. In an abstract sense, these pixels could be gray-scale values, RGB color values, or entire spectral curves. The focus of this work is to develop a practical synthesis framework that maintains this abstract view while synthesizing texture with high spectral dimension, effectively achieving *spectral invariance.*

The principle idea is to use a single monochrome texture synthesis step to capture the spatial information in a multispectral texture. The first step is to use a global color space transform to condense the spatial information in a sample texture into a principle luminance channel. Then, a monochrome texture synthesis step generates the corresponding principle band in the synthetic texture. This spatial information is then used to condition the generation of spectral information.

A number of variants of this general approach are introduced. The first uses a multiresolution transform to decompose the spatial information in the principle band into an equivalent scale/space representation. This information is encapsulated into a set of low order statistical constraints that are used to iteratively coerce white noise into the desired texture. The residual spectral information is then generated using a non-parametric Markov Random field model (MRF). The remaining variants use a non-parametric MRF to generate the spatial and spectral components simultaneously. In this approach, multispectral texture is "grown" from a seed region by sampling from the set of nearest neighbors in the sample texture as identified by a template matching procedure in the principle band.

The effectiveness of both algorithms is demonstrated on a number of texture examples ranging from greyscale to RGB textures, as well as 16, 22, 32 and 63 band spectral images. In addition to the standard visual test that predominates the literature, effort is made to quantify the accuracy of the synthesis using informative and effective metrics. These include first and second order statistical comparisons as well as statistical divergence tests.

2 Introduction

Texture synthesis has been an active topic in computer vision, image processing, and graphics. Initial work focused on monochrome texture images but has been adapted or extended to handle three channel, e.g. RGB, color texture as well. The idea of creating a general solution to the color texture problem has obvious theoretical value since textures are not limited to three color bands in the physical world. Recently, a more practical need has arisen due to the increase in popularity of multi/hyper-spectral imaging.

There is a need in the multi/hyper-spectral imaging community for robust and efficient techniques for creating unique synthetic data that exhibits known spectral/spatial characteristics. Multi/hyper-spectral imagery may be difficult or expensive to procure or in some cases may not exist at all. By synthesizing this imagery in the form of texture, algorithms designed for image segmentation, classification and target detection can be tested on novel backgrounds and scenes without the need to collect additional data. In a similar capacity, synthetic data is helpful in characterizing the performance capabilities of various optical systems through the use of computer simulation.

A number of synthetic background simulation systems are currently being used in the commercial sector, academia and government. The Missile Defense Agency (MDA) uses the Synthetic Scene Generation Model (SSGM) developed by Photon Research Inc., for generating ballistic missile simulations [43]. Another model developed at Photon Reaseach [27] is the GENESIS system for generating terrain simulations. The Digital Image Remote Sensing Image Generation (DIRSIG) model was developed at the Rochester Institute of Technology [17] and is used to generate synthetic imagery for a variety of applications. At the core of each of these simulation systems are large databases of spatial/spectral information used in the modeling process. Acquiring this data can be difficult and expensive. Since much of the information in these databases is in the form of material texture e.g. water, grass, trees, etc., texture synthesis may provide an attractive means for augmenting this data.

Unfortunately, even monochrome texture synthesis has proven to be a difficult task. A number of fundamentally different techniques have been explored for synthesizing monochrome texture. Of all the techniques found in the literature, arguably the most common approach is referred to as $synthesis - by - analysis$. These techniques rely on a statistical descrip-

tion of a texture based on estimates from a sample texture. The result is a statistical model that is meant to adequately and uniquely describe the characteristics of an ensemble of similar texture images. This ensemble is often illustrated as an infinite sheet of texture from which samples are drawn that although different, convey an overall sense of similarity. Some of the more specialized and obscure approaches have included techniques based on syntactic grammars [22, 35], reaction-diffusion and pde's [48], as well as iterated function systems (fractals) [4, 3].

Syntactic grammar techniques are similar to those found in natural language processing used for sentence parsing, noun phrase extraction and part-of-speech tagging. The idea is to create hierarchical placement rules governing the association of textural elements over a spatial extent. The major drawback of these techniques is that it is difficult to bootstrap the rules used to govern the placement of textural elements from a sample texture. In other words, the rules must be specialized, created a priori by hand.

The use of partial differential equations in image processing has proven very effective. By modelling images based on finite differences it is possible to describe the optical flow, level-set, and surface dynamics of an image. A particular technique known as reaction-diffusion (RD) uses pde's to describe the interaction between two processes that diffuse or propogate spatially at different rates. The slower process is said to diffuse while the faster is said to react, hence the name. Originally an idea of Alan Turing [45], the methodology has been adapted to describe a number of biological and physical processes. In the context of texture synthesis, the idea is that the faster process creates the low frequency information i.e. large-scale features, and the slower process produces the high frequency or small-scale features. Although these methods produce interesting results, it is difficult to apply these techniques across an arbitrary class of textures. The problem is that it is difficult to analyze a given texture and create the appropriate RD model.

A similar problem is found using fractals. Fractals operate on the principle of self-similarity. Through a process of recursive affine transformations an initial image is transformed into a new image that exhibits characteristic features across all scales. These recursive affine transformations are referred to as iterated function systems. A famous example is the fractal fern leaf that can amazingly be created beginning with an image of an arbitrary text document [36]. Although they can produce visually stunning textures under the right conditions, there lacks an effective means for analyzing an arbitrary texture and describing it as an iterated function system.

The mention of these techniques is simply for completeness. Although these techniques have both interesting and important theoretical value they are fundamentally different from the domain of *statistical texture modelling.* Since statistical texture modelling is the focus of this work it is important to introduce texture synthesis in the framework of statistical texture modelling.

2.1 Texture Synthesis as Statistical Texture Modelling

Representing a class of textures using a statistical description is formalized in the context of a random process or the discrete time equivalent, random sequences. A random process is defined as a random variable that changes continuously over time or space. Random processes have the property that they look exactly like a random variable at a particular time or location. As such, random processes may be sampled in a manner similar to sampling an ordinary distribution to create a *realization* of the process. For example, a realization of a random process (sequence) may include the price fluctuations of a particular stock over time, or an actual texture image.

A realization of a sample texture can be obtained by sampling from an ensemble of texture images. An ensemble is a set of functions with associated probabilities. A random process generating texture defines an ensemble over all possible texture images where those with similar textural qualities have a high probability of occurring.

There is a strong analogy between with a random variable and a random process. Without delving too deeply into the theory of random processes, a simple definition is that a random process simply maps random events onto a function whereas a random variable maps events onto the real line. To further illustrate, while a random variable may map an event to the number 5, a random process may map the same event to an image. In some sense then, a set of images generated by this random process can be used to characterize an underlying distribution in much the same way that multiple observations of a random variable may provide information about its distribution. The task of statistical modelling is then to discover the statistical properties of the underlying process in order to create novel realizations of the process. This is typically done by analyzing a finite sample of texture chosen to be representative of the entire ensemble.

2.1.1 Locality, Stationarity, Ergodicity and Homogeneous Texture Characteristics

In order to reliably and accurately estimate a statistical model from a finite texture sample, the sample must display the properties of locality, stationarity and ergodicity. Locality suggests that the spatial dependencies between different locations in a texture are local in nature. In other words, spatial correlation is generally highest over short distances. An addition property referred to as stationarity suggests that these local dependencies are shift invariant. The concept of stationarity is fundamental to the study of random processes. A stationary random process has the characteristic that correlation between events is solely a function of relative distance or offset. In other words, one never needs to know the exact location of two events, e.g. two pixels, just the relative distance in order to determine correlation. The practical significance of this is that correlation is periodic over an infinite extent of texture, meaning large areas of texture must contain redundant information. It is this redundancy that makes it possible to characterize an infinite ensemble of texture with a relatively compact yet complete statistical description.

Another important concept that applies to random processes is ergodicity. An ergodic random process is characterized by the fact that sample averages approach ensemble averages in the limit of infinite sample size. Short of this theoretical limit, the property of *practical ergodicity* as it is sometimes presented in the literature, suggests that a finite realization of a stationary random process provides a close approximation to the ensemble averages of the underlying process. Obviously if these assumptions hold, it would suggest that a texture sample would display all the relevant spatial correlations to define the perceptual characteristics of the entire texture ensemble. Hence for texture images created by a stationary ergodic random process, an appropriate statistical model should be able to generate the entire ensemble from a single finite example of texture. Textures that fall into this category are referred to as homogeneous. Homogeneous textures are further characterized by their tendency to be either structured or stochastic. A structured texture is a regular pattern e.g. a brick wall, whereas a stochastic structure is more random, e.g. grass. A heterogenous texture, which is arguably not a texture at all, would be something like an image of a human face. In general, a majority of the work done in texture synthesis has focused on the statistical modeling of homogeneous textures.

2.1.2 Markov Models and the Markov Assumption

A particularly useful tool for exploring statistical texture modelling of homogenous textures is the Markov Random Field model (MRF). An MRF is a 2D generalization of a 1D Markov chain. Markov chains condition the occurrence of an event based on observations from the past (or future). The parameters of a Markov chain model are a set of conditional probabilities describing the likelihood of an event based on n previous events. The validity of the model is dependent on the Markov assumption. It states that the occurrence of an event is dependent on at most n previous events, beyond which statistical independence is assumed. This suggests that there is a finite window over which things in the past can influence future events. The term "chain" refers to the unidirectional conditioning of discrete events that usually occurs in a left to right fashion over time or space. In other words, the occurrence of an event depends on the events directly to its left. Given the one sided nature of the model, Markov chains are said to be causal. To use a Markov chain to generate likely sequences of events, one simply multiplies the conditional occurrence of events at each step to form a product approximation of the joint probability over the chain.

$$P(\vec{x}) = P(x_0)\prod P(x_i|x_{i-1})$$

A number of efficient search algorithms can maximize this joint probability over a sequence of events.

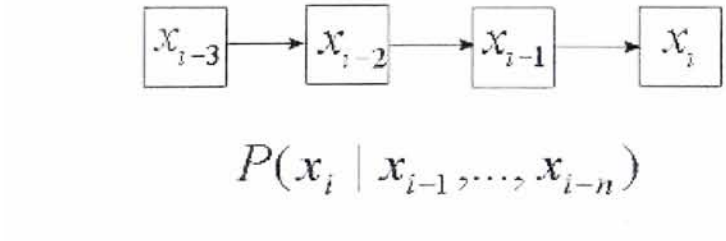


Figure 1: Illustration of a Markov Chain.

As a concrete example consider the modelling of stock prices. In this case, the goal is to guess the stock price at a particular moment in the future, based on information from the past. Using historic market data, the parameters of a Markov chain model are estimated. The model is then used to determine the most likely outcome of future events by finding the sequence of events that maximizes the product of the estimated conditional probabilities.

A challenging complexity is encountered when generalizing 1D Markov chains to 2D Markov Random Fields. The parameters of a Markov Random Field are conditional probability estimates for an event based on those surrounding it. In this case there is no longer a causal restriction on the nature of influence between events. Consider the occurrence of pixels on a 2D grid. Provided that the occurrence of a pixel across the grid is not statistically independent from every other pixel, the neighbors of a central pixel location provide information as to its value. As a result, influence must be considered from pixels in all directions. This means that there exists mutual and simultaneous influence between an event and those surrounding it spatially. This creates a chicken or an egg scenario where it is impossible to determine causality.

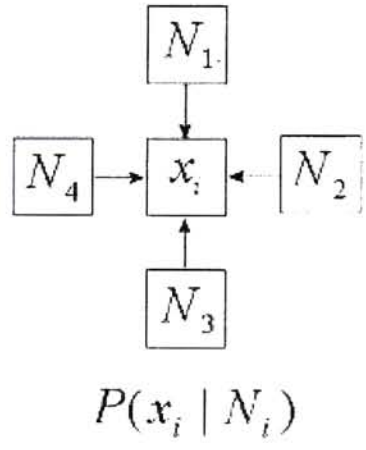


Figure 2: Illustration of the non-causal nature of the random field.

Because of this problem, algorithms for sampling a Markov Random Field involve combinatorial solutions over the space of all $n^2!$ ways of determining

causality over an nxn field. Practically speaking, using an MRF that directly estimates local probability density estimates in order to synthesize texture is a difficult task. Although a number of researchers have tried [29, 24], most of their work was done on massively parellel architectures which still took an inordinate amount of time to converge. However, from a theoretical standpoint, MRFs are invaluable for illustrating the complexities of statistical texture modelling. In fact, as will be illustrated in the next section, a number of successful synthesis methods rely on the formal assumptions of random field theory to implicitly estimate an MRF as part of their synthesis framework. It is precisely these techniques that are adapted into this work.

2.2 Previous Work On Monochrome Texture Synthesis Using Statistical Models

Most of the texture synthesis literature points to the work done by Julesz as the first effort to characterize the properties of texture, which led to the Julesz conjecture [31]. Simply put, the Julesz conjecture states that all the relevant spatial information in monochromatic texture images is contained in the first nth order joint statistics defined over local context neighborhoods of a texture. In this case *order* refers to the dimensionality of the pdf that describes the joint empirical pixel distribution. Of course there are practical limitations to how well the nth order joint statistics can be estimated as n gets large given finite sample data. Often referred to as the curse of dimensionality, the amount of data required to accurately estimate a joint pdf increases factorially with respect to dimensionality.

Early work using linear filters attempted to synthesize texture by imposing first and second order statistics alone. Cadzow [9] used a signal enhanced version of a texture sample to drive input noise followed by a histogram modification to create texture. The signal enhanced texture was used as the exemplar for the texture ensemble. By filtering white noise, random variation is added to each synthetic texture. A histogram modification scales the marginal statistics to match the exemplar texture.

In Cadzow's work, the signal enhanced exemplar forms a moving average (MA) finite impulse response filter (FIR). An alternative approach is to use ordinary least squares to create a autoregression (AR) filter. Deguchi and Morishita [15] used the grey level values in a local spatial neighborhood to calculate a regression model used to form a digital filter. Work done

by Gagalowicz et. al. [24] suggests that an approach similar to that taken by Cadzow using second order statistics gave better synthesis results. A combination of methods known as an autoregressive moving-average model (ARMA) was used by Strenzwilk et. al. [44] for performing texture synthesis.

More recent work in psychophysics has given rise to a new class of statistical texture models. Heeger and Bergen [30] used linear basis functions based on pychophysical models of human vision to decompose an image into a multiresolution representation. By iteratively matching marginal statistics independently over a set of oriented/scaled linear basis functions, they were able to synthesize a number of stochastic i.e. random textures. Unfortunately, textures with pronounced large-scale periodic structures proved difficult for this approach.

Work done by Portilla and Simoncelli [39] can be viewed as a direct extension of the Heeger and Bergen algorithm. Rather than simply match marginal statistics in each band independently, they enforced cross-scale and cross-band correlations as well as autocorrelation at each scale. A later version using complex analytic filters showed excellent results on a variety of textures [40].

Debonet and Viola [13] estimated a non-parametric Markov Random Field based on the hierarchal tree structure of a multiresolution representation. The algorithm progresses in a coarse to fine fashion filling in the levels in a synthesis pyramid using a heuristic search over the space of oriented band pass filter responses to a sample image. Similar neighborhoods where found forming a kNN distribution from which new pixels where drawn randomly.

Other methods using a similar nearest neighbor search can be found in the literature. Efros and Lueng [21] used an nxn neighborhood to select similar regions directly from a sample image, from which a central pixel location is copied to the synthetic texture. This method, although simple and effective, is extremely slow. As an interesting anecdote, in 1981 Garber [25] actually explored an identical approach but decided that it was too costly for a practical implementation. This approach also has problems generalizing since an appropriately sized neighborhood has to be selected for each texture example so as to capture the extent of the largest spatial feature.

An alternative solution combining the work of Debonet and Viola with Efros and Lueng was proposed by Wei and Levoy [47]. Using a multiresolution decomposition and a small fixed sized neighborhood across scales, they were able to create a very robust representation that generalized well across

textures exhibiting textural features with highly variable spatial frequencies. This removed the need for human operator input to determine an appropriately sized neighborhood. As with the Debonet and Viola method, the hierarchical structure imposed on the search space allowed them to generate good quality texture relatively quickly. An additional optimization was achieved by using a tree structured vector quantization step to impose a quantized hierarchy on the search space. The result was that an approximate nearest neighbor search was used that's not guaranteed to be globally optimal but was shown to be successful across a wide range of textures.

Non-parametric search strategies like those described have been referred to as *resampling* in the literature. Although effective and modified to be efficient, at times resampling techniques exhibit stability issues when there aren't enough similar regions or when they become stuck in one region of the search space. Under these conditions resampling techniques have been shown to grow uncharacteristic texture [47].

A novel solution has recently been developed by a number of independent sources [33, 20]. The technique is commonly referred to as image quilting and is similar to pixel-by-pixel resampling except, as the name implies, whole regions of a sample image are copied at each step. Which region to be copied is generally determined by the similarity of border overlap with other regions in a sample image. Quilting has emerged as a fast and powerful technique for synthesizing a wide range of texture types.

2.3 Previous Work Extending Monochrome Synthesis to Multiple Bands

Color texture synthesis has been handled in a number of ways. Heeger and Bergen extended their technique by performing a PCA transform to decorrelate color channels and then synthesize three independent textures and backproject. Liang et. al. [34] demonstrated the problem with this approach by illustrating the fact that textures have "colorful edges" which likely result in responses across bands. In this way they demonstrated that cross-band correlations are largely local in nature and cannot be removed by a global color space transform. Instead they used a Gaussian Scale Mixture model to cancel local variance scaling in the residual bands of an ICA transformed image. Good color synthesis results were obtained by simply synthesizing in the dominant luminance channel i.e. the channel carrying the most spatial

information, and then modulating it with a pair of variance scaled residual bands.

Work on multispectral texture has been sparse in the literature. Botchko et. al. [7] used a parametric model of multispectral texture to perform synthesis-by-analysis. The spatial correlation of an input texture was modelled as a Gaussian Markov Random Field estimated by the power spectrum of a sample image. The spatial information was generated using a sum of two-dimensional sinusoids where the frequency was sampled from the power spectrum with additional stochastic parameters corresponding to phase shift and magnitude. Obviously such a model introduces a tremendous amount of bias, in other words a great deal of the information in the resulting image is introduced by the model. An additional assumption was that spatial/spectral information is separable. Although work has been done showing this is a reasonable assumption in natural images, the model was too restrictive in that a small class of parametric functions where chosen to capture spectral correlation.

Dodd [18] used a similar approach to that of Heeger and Bergen by using a PCA transform to decorrelate the color space of multispectral components. As already demonstrated this assumption may not always be valid. Further analysis in this work will prove conclusively that a global color space transform cannot be used to reliably separate spatial and spectral processing.

3 Exploring Monochrome Texture Synthesis Techniques

It would be nice to identify methods that are efficient in both time and space, generalize well over a variety of texture types, require a minimum amount of human operator input, and give high quality synthesis results. The ability of a texture synthesis algorithm to meet most of these requirements is relatively easy to determine. Unfortunately, this doesn't include a quantitative determination of how well a texture can potentially or actually be synthesized. Consider the fact that texture synthesis techniques commonly use some type of metric to determine similarity between input and output in order to drive the synthesis process. Using the Portilla and Simoncelli method as a model it should be possible to enforce virtually any similarity metric as a constraint in their synthesis framework. An additional complication is that a complete set

of metrics that quantitatively define the effectiveness of a synthesis procedure is an open problem and constitutes much of the work done in developing texture synthesis algorithms. In addition to synthesis quality, additional factors including robustness, stability and speed need to be considered. As a first order approximation, an empirical evaluation of common synthesis techniques is a logical first step.

3.1 Resampling

The first method to be implemented and tested is pixel-by-pixel resampling. This technique is simple and results in the literature are good. The method works by seeding a synthetic image with a small patch of texture taken from an example image. The remaining synthetic texture is generated a single pixel at a time proceeding in a circular fashion around the seed region. At each step a local context is defined by the neighboring pixels of the current pixel to be synthesized. The size of the local context is user specified and should capture the largest extent of repeating structures. The local context becomes an exemplar template that is used to search the sample image. Generally a Euclidean distance metric is used but a normalized cosine similarity is also effective:

$$\begin{aligned} z_{i,j} &= \sum (S_{i,j} - T_i)^2 \\ z_{i,j} &= (S_{i,j} * T_i)/|S||T| \end{aligned}$$

where z is *distance*, $S_{i,j}$ is the i,jth pixel in a sample image and T is the exemplar template. A threshold on $z_{i,j}$ is used to determine a set of similar regions in the sample image. This set defines a non-parametric Markov Random Field that is then sampled at random to produce the current pixel value. This process continues until all pixels in the synthetic image have been synthesized. As mentioned, this process is extremely slow. One experiment using a 128x128 sample image with a 16x16 neighborhood to generate a 256x256 synthetic image, took over 24 hours to complete on a desktop Pentium system running at 750 mhz. Further work using optimization methods have shown to greatly improve the speed of this algorithm and will serve as a fundamental component of the algorithms developed later in this work.

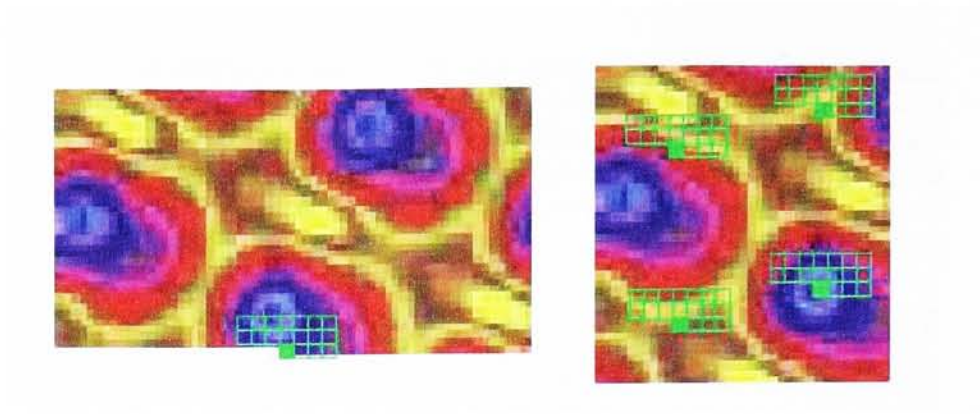


Figure 3: Pixel-by-Pixel resampling illustrated.

3.2 Multiresolution Resampling

Debonet and Viola's [13] multiresolution technique relies on similar principles used in pixel-by-pixel resampling, but promises to be much faster and more robust. The key to this gain in efficacy is that a multiresolution representation is used to analyze a sample texture. A multiresolution representation provides both scale and space information at each location in an image. Essentially, a multiresolution representation provides information about a pixel's neighbors at different scales. At the finest scale there might be information about the difference between each pixel and its immediate neighbors. Progressing to coarser scales, information from larger and larger spatial extents will be encoded in the representation. This type of information is useful for texture synthesis in a resampling framework for a number of reasons. First, rather than capturing local spatial context using a fixed scale neighborhood as before, a multiresolution representation can provide spatial information across all scales simultaneously. This means there is no longer any need for a human operator to choose an optimal neighborhood size based on the current texture. Not only has the necessity for human involvement been eliminated, but a potential increase in robustness is introduced. Under the traditional resampling paradigm a single fixed sized neighborhood is selected to capture local spatial context at the most appropriate scale. It is certainly conceivable that a particular texture might be very complicated, exhibiting a number of pronounced spatial correlations at widely different scales. Choosing a single analyzing neighborhood might be too rigid an ap-

proach that doesn't capture the essence of the texture's spatial correlation. Under a mutliresolution framework this problem may be alleviated since it is possible to extract information about spatial context from all possible scales from a sample texture.

3.2.1 Laplacian Pyramid

Described simply, a Laplacian pyramid is a multiresolution decomposition of an image created by iteratively low passing a image with a Gaussian filter and then taking the difference between filter responses. The net effect is a multi-level, band passed representation of the original signal using a difference of Gaussian (DoG) or Laplacian operator. Each low pass filtering operation is performed such that the resulting filter response is approximately band limited by a factor of two in each direction. The sampling theorem says that in this case half the coefficients in each direction will be redundant and can be removed. These coefficients are removed by downsampling at each stage of the iteration. Hence, when each band passed response is ordered from coarsest to finest, a hierarchical pyramid is formed Figure 4. The height of the Laplacian pyramid is limited to $\lfloor \log(n) \rfloor$ levels where n is the smaller of images width or height.

In order to determine the wavelet coefficients at a particular scale, the signal is low-passed through a Gaussian filter, then the difference is formed between the current and low-pass versions of the input. The result is the determination of a residual high-pass response to the input that, when inserted into an iterative framework, becomes a band-pass response to the original signal. In this case, the high pass filter in the filter pair is implicitly defined by the differencing operation between the low-pass and original signal. The Laplacian pyramid basis functions are easily extendable from 1D to 2D using a separable Gaussian kernel.

Reconstruction of the original signal from its Laplacian pyramidal decomposition is quite simple. Starting with the dc term (or the lowest scale in the pyramid), up sample and filter with the same Gaussian kernel used in the forward operation. Next, add the band pass information at the next highest scale. Repeat this process until the last band is added to the signal.

An interesting property of the Laplacian pyramid is that of *overcompleteness*. Overcompleteness refers to the fact that the wavelet filters are shifted in the spatial or time domain in such a way that they overlap. The property of overcompleteness implies redundancy in the wavelet coefficients which af-

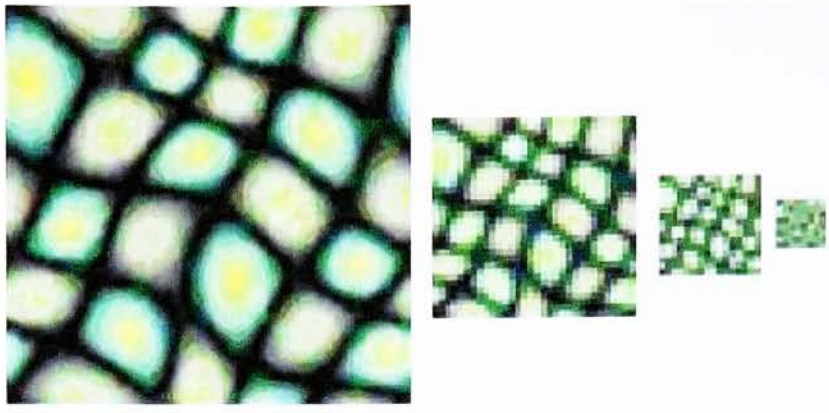


Figure 4: The results of an iterative application of a DoG filter with downsampling.

fects the invertibility of the forward transform. In short, any redundancy suggests that the multiresolution transform cannot be inverted perfectly.

Recalling Parseval's Theorem, a condition for the existence of the inverse Fourier Transform is that the energy in both the spatial and spectral representations of a signal must be equivalent. When viewed as a simple transformation of variables, i.e. the discrete case, this relationship suggests that the spatial and spectral representations must have the same length and dimension.

In the case of the Laplacian pyramid, the number of wavelet coefficients outnumbers the pixels by a factor of 4/3. In this sense, the Laplacian pyramid is said to be 4/3 overcomplete. To illustrate, starting with a 64x64 image there are 4096 samples. In the corresponding Laplacian pyramid representation there are 5460 + 1 coefficients giving an overcompleteness factor of 4/3 (the + 1 refers to the low-pass residual). This overcompleteness is typical in a number of multiresolution transforms and is directly related to the fact that the high-pass residual is not downsampled. Consider what happens at the bottom level (ie finest scale) of a Laplacian pyramid. The total number of coefficients is equal to that of the incoming signal. Any additional coefficients at the higher scales will result in overcompleteness.

Formally, the basis functions for the Laplacian pyramid do not form a *tight frame*. When a wavelet basis can perfectly reconstruct a signal it is referred

to as a *frame*. The following condition must be satisfied for a wavelet basis to form a frame.

$$A \|f\|^2 \leq \sum_{j,k} |\langle f, \Psi_{j,k} \rangle|^2 \leq B \|f\|^2$$

This condition states that the energy in the wavelet coefficients is bounded by two constants A, B. When $A == B$, the resulting basis is called a *tight frame*. Tight frames act like an orthonormal basis, meaning the analyzing and reconstructing wavelets are complex conjugates of each other. This is similar to the Fourier transform. What's further, when $A == B == 1$ the tight frame is actually an orthonormal basis.

As is discussed in Appendix A, the fact that the Laplacian pyramid is not perfectly invertible is tolerated given that the reconstruction error is generally small for well behaved signals. Also, some advantages are realized when using a overcomplete representation. Specifically, overcomplete represenations can be made quasi shift-invariant which is invaluable when analyzing the spatial correlation in a texture which is assumed to stationary. Refer to Appendix A for a tutorial on wavelet and multiresolution techniques.

3.2.2 Steerable Pyramid

The steerable pyramid is a straightforward extension of the Laplacian pyramid representation just described. After a single filtering step used to extract the high frequency residual information, the image is again passed through a series of low-pass filters. The difference this time is that the band passed coefficients at each level of the pyramid are further decomposed using a set of oriented filters Figure 5. These oriented filters are generally rotated versions of a two-dimensional second derivative operator Figure 6. The result is a steerable pyramid representation describing the oriented spatial information at each scale of the original image.

This means that instead of a single set of coefficients at each scale, there is a separate set of coefficients corresponding to the responses of each oriented filter Figure 7. A steerable pyramid is overcomplete by a factor of $k4/3$ where k is number of oriented filters used. It is also shift and rotation invariant. Additionally, the steerable pyramid is quasi-invertible with errors generally acceptable for most graphics applications.

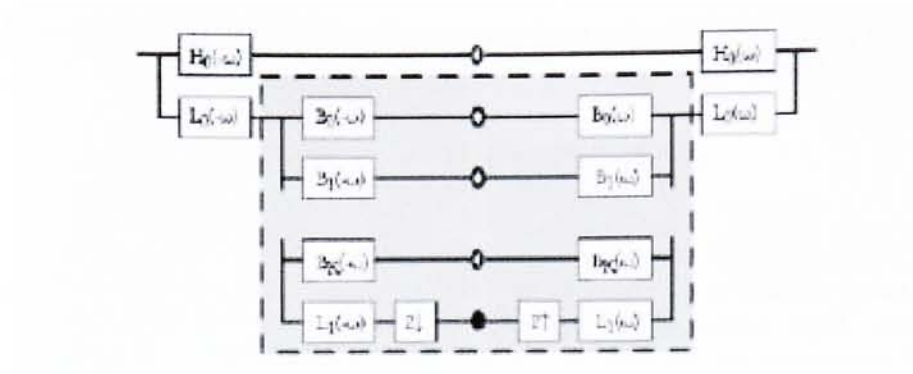


Figure 5: Diagram showing application of filter bank in a steerable pyramid framework [40].

The purpose of the steerable pyramid is to capture local oriented structure at each scale in the decomposition. So, it is natural to say that the purpose of the steerable pyramid is used to extract meaningful features from a signal. This is an example of how powerful analysis tools can be constructed within the wavelet or multiresolution framework. Refer to Appendix A for a more complete description.

3.2.3 Nearest Neighbor Search Over a Multiresolution Feature Space

Multiresolution pyramid decompositions form a natural hierarchical dependency between coefficients at adjacent scales. Due to the downsampling and averaging operations performed when building a pyramid, each coefficient at one level closely approximates the average value of a group of four coefficients in the next finest scale. By selecting a coefficient from each level so as to preserve this dependency, feature vectors can be extracted from the pyramid that encapsulate the scale/space information at a pixel location in the original image. These feature vectors can then be used to create an informative feature space over which a nearest neighbor search can be performed.

This is the approach taken by Debonet and Viola [13]. First, a steerable pyramid representation of a sample image referred to as the *analysis side* is generated. Then, a *synthesis side* steerable pyramid with the same number of levels as the input texture pyramid is initialized using a noise image. Then

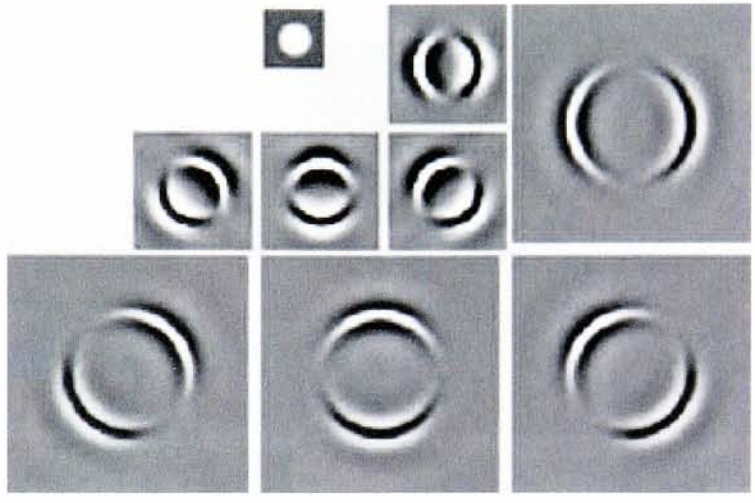


Figure 6: Diagram showing steerable filter bank [40].

proceeding in a course-to-fine fashion, the synthesis side pyramid is populated level by level with coefficients taken from the input side pyramid. These coefficients are determined via a nearest neighbor search over the parent structures previously described Figure 8.

Since the highest levels of each pyramid don't have a parent, an initial step is needed to start the searching process. Unfortunately this is where the algorithm shows its weakness. Generally, the top level of the analysis side pyramid is a single pixel whereas the synthesis side will be larger. So as an initial step the single value, i.e. dc term, from the analysis side is copied to the synthesis side. This means that each pixel in the next finest level all share the same parent. The result is a redundant context that causes tiling at the coarsest levels in the pyramid. In fact, the top two levels of the synthesis pyramid are always predetermined using this method. This may be an advantage when dealing with highly structured textures but can adversely effect the algorithm's ability to handle more random structures.

Aside from this shortcoming, an additional benefit of using a multiresolution representation is that they can greatly improve the speed at which texture can be synthesized. Due to the fact that a multiresolution pyramid imposes a hierarchical structure on the space of similar neighborhoods, it is possible to determine unfruitful search paths relatively quickly. By in-

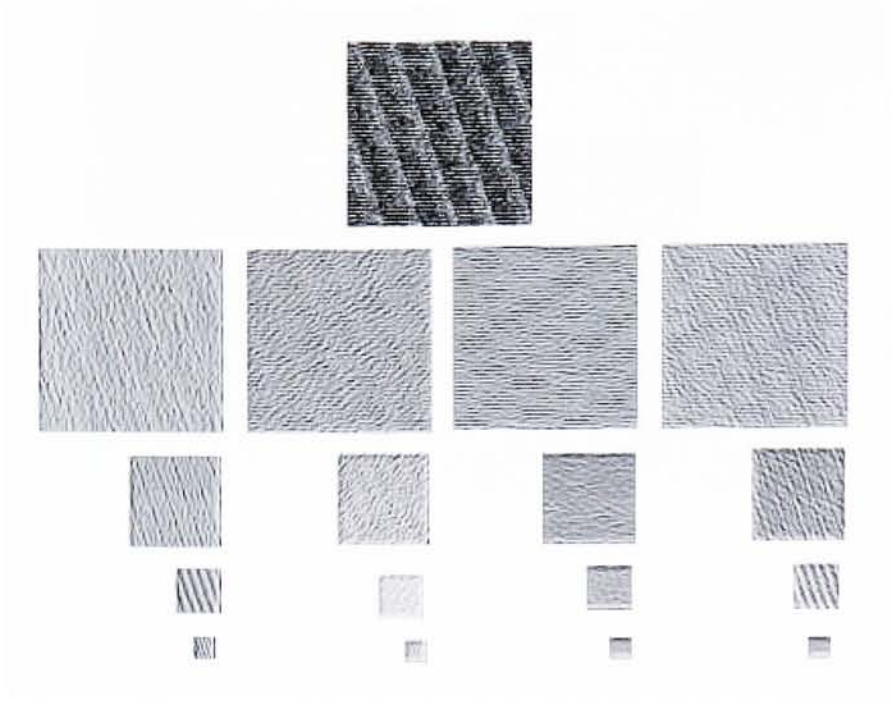


Figure 7: Steerable pyramid decomposition of image on top.

corporating the appropriate logic into a heuristic search the task of finding nearest neighbors can be made much more efficient. As a comparison the same experiment using pixel-by-pixel resampling that took over 24 hours, finished in less than 5 minutes using the Debonet method. Clearly this is an improvement in time and as pointed out it's reasonable to assume that this method is more robust. Unfortunately, resampling methods are inherently prone to failures due to the searching process. Each step in a resampling process is "greedy" providing no means of backtracking from an erroneous step. The result is that synthesis failures often times degrade quickly, essentially substituting a single error into a feedback loop that causes serious stability problems. What's needed is a way to control synthesis failures so that some statistical properties can be ensured even if the perceptual aspects of the texture are incorrect.

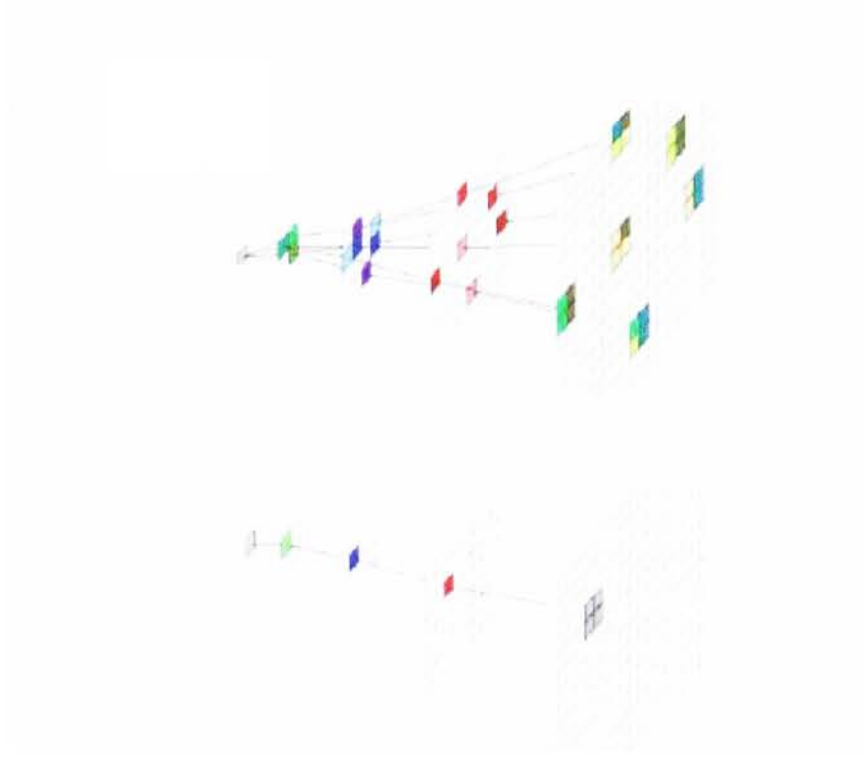


Figure 8: Hierarchical structure of a Laplacian pyramid used to identify similar neighborhoods [13].

3.3 Moving Average Filters

The Wiener-Kinchen theorem provides a means to enforce the autocorrelation of a function onto white noise through the use of a linear filter.

$$\begin{aligned} I &= GWN(0,1) \star f \\ I \star I &= GWN(0,1) \star f \star GWN(0,1) \star f \\ R_{I,I} &= \delta \star f \star f = R_{f,f} \end{aligned}$$

Here $GWN(0,1)$ represents Gaussian White Noise with zero mean and unit variance, I is the resulting texture, f is the sample texture and R is

the correlation function. This is a fundamentally different approach than the previous methods. By using an example image as a linear filter driven by zero mean, unit variance noise, it is possible to synthesize texture with known statistical properties. This is a significant departure from the graphics based approaches based on resampling. The enforcement of statistical constraints provides a rigorous framework for synthesis-by-analysis techniques. The advantages are obvious since constraint enforcement can provide a stop gap measure to handle synthesis failures. In other words, regardless of visual appearance, the target statistical constraints are still guaranteed to be enforced.

A finite impulse response filter (FIR) has finite extent, and is used in a simple convolution operation. This type of filter is referred to as *moving average* (MA) since it has the effect of averaging values over a finite window. In order to synthesize texture, simply use a sample image as a moving average filter convolved with white noise. By performing the convolution in the Fourier domain, the entire synthesis operation can be performed in seconds. After a mean and variance scaling operation, the resulting texture will have the same first and second order statistics as the input texture. Experimental results suggest that this technique is able to create synthetic texture with reasonable visual appearance.

3.4 Portilla and Simoncelli Method

The Portilla and Simoncelli method (P/S) [40] is a synthesis-by-analysis technique that enforces a set of statistical constraints over the output of a complex analytic filter bank. Like the moving average technique, the synthesis results are guaranteed to display target statistical characteristics regardless of actual visual appearance. In this case, the statistical constraint set is richer and when applied over a multiresolution representation provides better spatial fidelity. The process is iterative, proceeding in a coarse-to-fine fashion over the pyramid. Using alternating projections over a statistical constraint set, the P/S method coerces white noise into the desired output. The method is highly effective over a broad range of texture types, has good time/space efficiency and requires no human operator input (certain aspects of the algorithm are tunable if desired).

3.4.1 Complex Steerable Pyramid

The Portilla and Simoncelli method uses a variant of the steerable pyramid. The linear filters in this transform are an extension of the oriented derivative operators used by Debonet Figure 6. In this transform, the band pass filters are oriented versions of a common function that form a complex Hilbert Transform pair:

$$
\begin{aligned}
B_k(r,\theta) &= H(r)G_k(\theta) \\
H(r) &= \begin{cases} cos\left(\frac{\pi}{2}log_2\left(\frac{2r}{\pi}\right)\right) & , \quad \frac{\pi}{4} < r < \frac{\pi}{2} \\ 1 & , \quad r > \frac{\pi}{2} \\ 0 & , \quad else \end{cases} \\
G_k(\theta) &= \begin{cases} \alpha_k \left[cos\left(\theta - \frac{\pi k}{K}\right)\right]^{K-1} & , \quad \left|\theta - \frac{\pi k}{K}\right| < \frac{\pi}{2} \\ 0 & , \quad else \end{cases}
\end{aligned}
$$

Here, $\alpha_k = 2^{K-1}\frac{(K-1)!}{\sqrt{K(2K-1)!}}$ where K is the number of orientations used, in this case four. In addition, there is a single low pass filter:

$$
L(r,\theta) = \begin{cases} 2cos\left(\frac{\pi}{2}log_2\left(\frac{4r}{\pi}\right)\right) & , \quad \frac{\pi}{4} < r < \frac{\pi}{2} \\ 2 & , \quad r < \frac{\pi}{4} \\ 0 & , \quad else \end{cases}
$$

Empirical results found by Portilla and Simoncelli suggested that a number of synthesis failures could be attributed to a failure to correctly represent local phase information in a texture. By using complex analytic filters, local phase information can be used to detect the polarity of edge and boundary transitions hence alleviating the problem.

When analyzing signals it is sometimes useful to determine local phase shifts associated with filter responses over regions with edges or lines of different polarities e.g. dark line on a light background, or edge transition from dark to light. In order to capture this information in a multiresolution framework, it is possible to extend the steerable pyramid to include magnitude and phase information. This is done by replacing the ordinary directional operators with an even/odd filter pair corresponding to the real and imaginary parts of a signal Figures 9, 10. By keeping the pyramid overcomplete, the new transform remains shift and rotation invariant. Reconstruction is similar to that of the steerable pyramid except now the conjugated inverse filters are actually different since they contain an imaginary part.

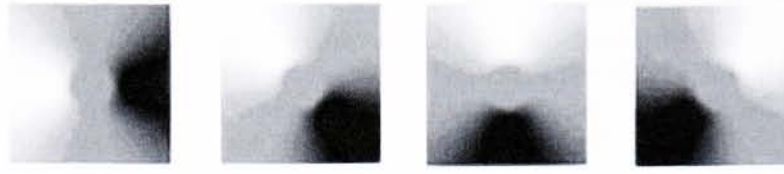


Figure 9: Real part of frequency responses of complex steerable filters at four orientations.

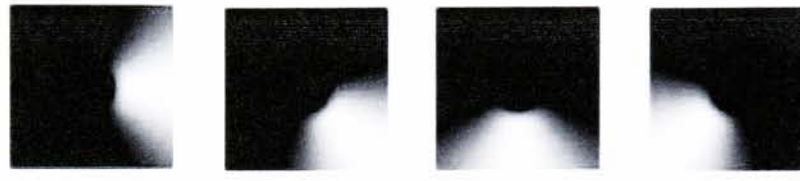


Figure 10: Imaginary part of frequency responses of complex steerable filters at four orientations.

Refer to Appendix A for more information on wavelets and multiresolution analysis.

3.4.2 Statistical Constraint Enforcement

Statistical constraint enforcement has the attractive property of producing results that display target statistical characteristics. The problem is to find a rich constraint set that captures the spatial information in a texture. Although an optimal constraint is difficult to determine for all texture types, empirical results suggest that the autocorrelation and marginal pixel distribution are important [39].

In addition to multi-scale autocorrelation, the Portilla and Simoncelli technique uses cross-scale and cross-band covariance corrections. These corrections are needed because the band oriented responses are highly correlated. This happens for two reasons, first is that the basis functions are overcomplete so they share information across orientations, secondly texture

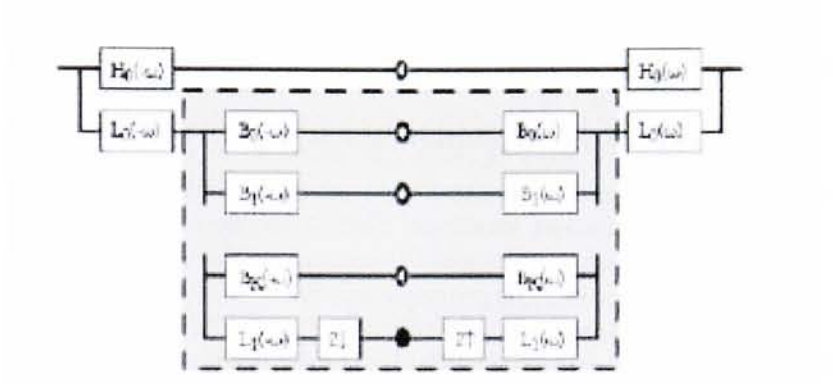


Figure 11: Flow diagram showing application of filters in P/S method [40].

features like edges and lines are likely to have responses at multiple scales and orientations.

The following is a summary of the statistical constraints used by the Portilla and Simoncelli method: 1) The first four pixel moments, mean, variance, skewness and kurtosis. 2) Subband autocorrelation of the N central samples. 3) Cross-Band magnitude covariances. 4) Cross-Scale, phase-doubled magnitude and real covariances. Refer to Figure 12 for an overview of the algorithm.

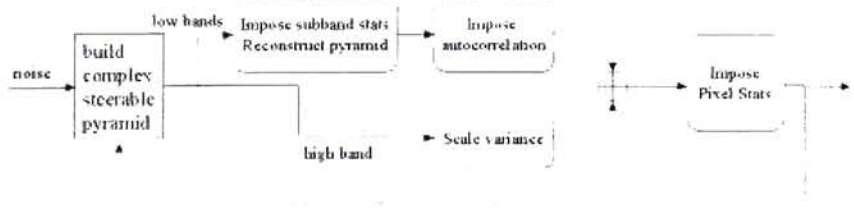


Figure 12: Flow diagram of the P/S algorithm.

For this work a simple histogram modification is used to enforce the marginal pixel statistics. The autocorrelation adjustment is similar to the moving average adjustment previously introduced. The last two constraints

are based on linear projections.

3.4.3 Adjusting Autocorrelation Statistics

An image's autocorrelation is the lowest order moment that describes dependency between adjacent locations. Similar to the correlation coefficient encountered when dealing with pairs of random variables, the autocorrelation of a random process or random sequence is a function of such coefficients (although they are unnormalized and therefore not constrained to the interval [-1,1]). Since the autocorrelation contains spatial correlation, the adjustment of the autocorrelation statistics is a natural choice in a synthesis framework.

Similar to the moving average filter operation used by Cadzow [9], the autocorrelation is enforced at each scale of a pyramidal decomposition. The difference this time is that the input is no longer white noise. Instead, it represents the current state of the iterative constraint enforcement process and is therefore a coherent signal. In other words, the input image itself has non-trivial autocorrelation. So, in order to enforce autocorrelation, the filter is no longer the actual target image. Instead, a filter h is found so as to enforce a target autocorrelation on a coherent image:

$$\begin{aligned} R_T &= I \star h \star I \star h \\ R_T &= R_h \star R_I \end{aligned}$$

or in the frequency domain,

$$S_T = |H|^2 S_I$$

In this case the filter h is found quite simply to be:

$$h = \mathcal{F}^{-1}\{+\sqrt{S_T/S_I}\}$$

3.4.4 Adjusting Sub-band and Cross-band Covariance Statistics

As mentioned, the filter responses at each scale are highly correlated. In addition, the filter responses at adjacent scales are also highly correlated. Capturing this correlation is important for ensuring good quality synthesis results. In this work, covariance statistics are used to capture this correlation.

The enforcement of covariance statistics is straightforward in both the sub-band and cross-band cases.

To begin with, consider enforcing the sub-band covariance statistics across the K (in this case $K = 4$) filter responses at each scale. In order to adjust covariance statistics we seek a matrix M that can enforce a particular covariance over a set of data.

$$\begin{aligned} \boldsymbol{C_T} &= \boldsymbol{E\{MXX^TM^T\}} \\ \boldsymbol{C_T} &= \boldsymbol{MC_XM^T} \end{aligned}$$

Here, $\mathbf{X}$ is a matrix of observations where each observation is a $1x4$ column vector containing the wavelet coefficients from each of four oriented filter responses. To illustrate further, suppose the filter response at a given scale are each of size $64x64$. Then, $\mathbf{X}$ will contain $64x64$ observations each of dimension $1x4$. Also, note that the resulting covariance matrix is of dimension $4x4$ since there are four filter responses at each scale of the pyramidal decomposition.

Writing the equation explicitly in terms of the samples in matrix $\mathbf{X}$, the covariance can be expressed as:

$$C_{n,m} = E\{x_n(i,j)x_m(i,j)\}$$

Here, n, m refer to the filter responses at a particular scale and in this case range from 1 to 4. The indices i, j refer to spatial location. Referring to the equations above, the matrix $\boldsymbol{M}$ needs to normalize the covariance of $\boldsymbol{X}$ and then enforce a new covariance:

$$\begin{aligned} \boldsymbol{X'} &= \boldsymbol{MX} \\ \boldsymbol{X'} &= \boldsymbol{V_tD_t^{1/2}D^{-1/2}V^TX} \end{aligned}$$

The result of this operation namely, $\boldsymbol{X'}$ satisfies the equation:

$$\boldsymbol{C' = E\{X'(X')^T\} = C_T}$$

The term, $\boldsymbol{VD^{-1/2}}$ is simply a whitening transform. Therefore, $\boldsymbol{V}$ is a set of eigenvectors and $\boldsymbol{D}$ is a diagonal matrix of singular values. The

subscript $\boldsymbol{t}$ denotes the term is derived from the target data. In other words, the subscript $\boldsymbol{t}$ refers to the statistics that are to be enforced. It is important to note that this adjustment is exact and is in fact an orthogonal projection operation.

The cross-band magnitude and phase doubled covariance statistics are enforced in a similar way. The only difference is that the covariance function is replaced by a cross-correlation function. In other words, rather than working with $\boldsymbol{E\{MXX^TM^T\}}$, the matrix $\boldsymbol{E\{MXY^TM^T\}}$ is used. In this case, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are samples derived from adjacent bands. These bands are naturally referred to as a mid and low band. Since the resulting matrix is still square, $\boldsymbol{4x4}$, the number of samples from each scale must be the same. However, since the band pass responses at adjacent scales differ in size by a factor of two, the filter responses in the low band are upsampled and interpolated prior to generating the cross-covariance statistics.
Beginning with the explicit expression for the cross-band covariance the necessary orthogonal projection can be derived.

$$B_{n,m} = E\{x_n(i,j)y_m(i,j)\}$$

By taking the partial derivatives of this expression with respect to x_n it is possible to construct an expression for the desired adjustment. Beginning with a general gradient projection of the form:

$$\overrightarrow{x}' = \overrightarrow{x} + \lambda_k \overrightarrow{\nabla}\theta_k(\overrightarrow{x})$$

Simply substitute $B_{n,m}$ for $\theta_k(\overrightarrow{x})$ in the above expression one arrives at the expression:

$$\overrightarrow{x_k}' = \overrightarrow{x_k} + \sum_{n=1}^{K} \theta_{k,n}\overrightarrow{x_n} + \sum_{m=1}^{K} \mu_{k,m}\overrightarrow{y_m}$$

This expression is more conveniently expressed in vector-matrix notation as:

$$\boldsymbol{X' = MX + KY}$$

As with the sub-band case, the matrix $\boldsymbol{M}$ is an unknown $\boldsymbol{4x4}$ matrix but in this case there is an additional $\boldsymbol{4x4}$ matrix $\boldsymbol{K}$ of unknown coefficients as

well. To solve for these two matrices note that the covariance of $\boldsymbol{X}'$ contains four terms:

$$\boldsymbol{E\{X'X'^T\} = MCM + MBK + KB^T + KEK^T}$$

Here, $\boldsymbol{C}$ is the covariance of $\boldsymbol{X}$ and $\boldsymbol{B}$ is the cross-scale covariance as defined previously, and $\boldsymbol{E}$ is the covariance of $\boldsymbol{Y}$. Now using the expression for the cross-covariance of $\boldsymbol{X}'$ and $\boldsymbol{Y}$, the final transform can be found in two steps.

$$\begin{aligned} \boldsymbol{E\{X'Y^T\}} &= \boldsymbol{ME\{XY^T\} + KE\{YY^T\}} \\ &= \boldsymbol{MB + KE} \end{aligned}$$

Setting this last equation to $\boldsymbol{B_t}$, i.e. the target cross-covariance, and solving for $\boldsymbol{K}$, a symmetric quadratic constraint is found of the form:

$$\boldsymbol{M[C - BE^{-1}B^T]M^T = C_t - B_tE^{-1}B_t^T}$$

But this equation is the same form as $\boldsymbol{MCM^T = C_t}$ which is exactly the form used to derive the sub-band covariance adjustment. The diagram in Figure 13 illustrates the sub-band and cross-band constraint enforcement operations. In this diagram, $\boldsymbol{\phi_i}$ represents the sub-band covariance statistics and $\boldsymbol{\omega_i}$ represents the cross-band covariance statistics.

The last aspect of the algorithm to consider is the way in which the statistical constraints are enforced. Unfortunately, there is no way to enforce the constraints simultaneously using linear projection methods. What's needed is an iterative search technique that is efficient and produces good results.

Maximum entropy constraint enforcement is a powerful technique for imposing a set of constraints subject to the condition that no other information is introduced. Maximum entropy works by minimizing the cross entropy between a uniform distribution and a distribution that satisfies the statistical constraints. By doing so, maximum entropy ensures no other information or bias is introduced into the model. Cross entropy is a statistical measure of how close two distributions are. An analogy is Euclidean distance between vectors in a metric space. The problem with maximum entropy methods is that they require iterative numerical solutions to find an appropriate distribution followed by combinatorial methods to sample the distribution to create a realization of the random process.

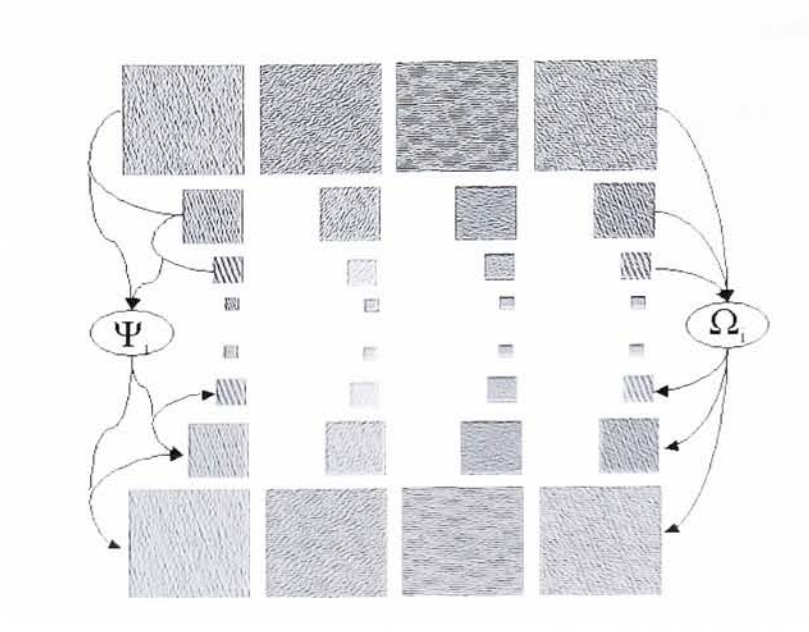


Figure 13: Illustrating the enforcement of the sub-band and cross-band covariances.

3.4.5 Projection Onto Convex Sets

Another approach is to use alternating projections. Alternating projections is derived from the principle of Projection onto Convex Sets (POCS). This principle states that a set of constraints can always be satisfied using iterative orthogonal projections provided that the functions satisfying each constraint define a convex set and there is a non-empty intersection over all convex sets. To illustrate this idea consider the following, all functions satisfying a particular constraint lie in a subspace in some n dimensional metric space. To be convex, the subspace must satisfy the following condition: $\forall p, q \in S, (a * p + (1 - a) * q \in S)$.

Since the goal in the P/S framework is to enforce a set of constraints

that obviously exist simultaneously, the condition of non-empty intersection is satisfied. After all the constraints themselves are all derived from a single source image. Unfortunately, the constraints Portilla and Simoncelli use are not convex. Even though convergence is not guaranteed, empirical results suggest alternating projections reaches a stable state relatively quickly for the aforementioned set of constraints. A final diagram shows a detailed breakdown of the sub-band and cross-band statistical constraint procedure Figure 14.

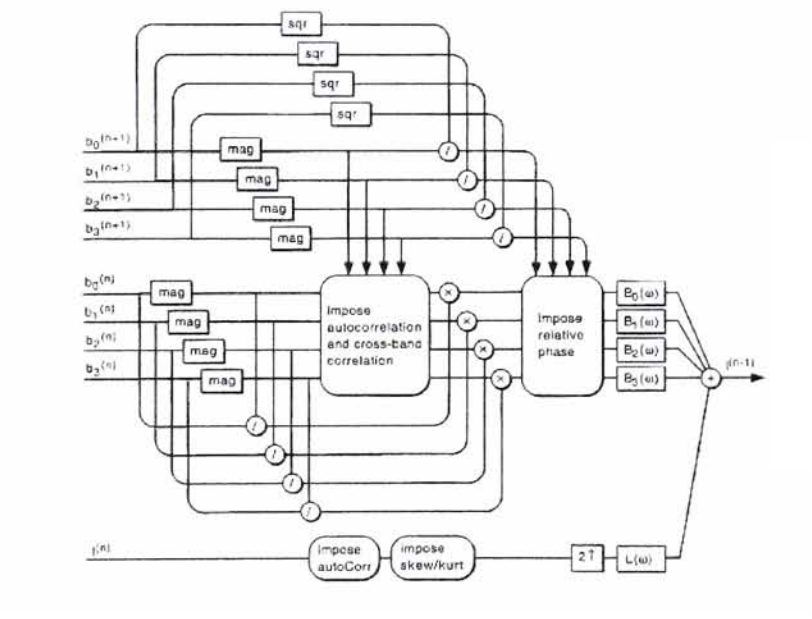


Figure 14: Sub-band and cross-band statistical constraint enforcement [40].

3.4.6 Psuedo-Code

In summary, a brief outline in psuedo-code illustrates the Portilla and Simoncelli monochrome texture synthesis technique.

```
N := Noise

P/S-TexGen(Sample,N,Orients,Levels,iterations)
    Pyr = BuildPyramid(Sample)
    Stats = GenStats(Pyr)

    for i = 1 to iterations
        PyrSynth = BuildPyramid(N,Orients,Levels)
        N = low(PyrSynth)
        for s = 1 to scales
            EnforceCrossBandCov(Stats,PyrSynth.Bands(S))
            If( S > 1 )
                EnforceCrossScaleCov(Stats,
                                     PyrSynth.Bands(S),
                                     PyrSynth.Bands(S-1)
                N = CollapseLevel(N,PyrSynth.Bands(s))

            EnforceAutoCorrelation(Stats,N)

        AddHiResidual(N,PyrSynth.Hi())
        MatchHistogram(N,Sample)
```

4 A New Direction

The focus of this work is to find a robust, efficient way to synthesize multispectral texture. When modelling texture with high spectral dimensionality, it becomes necessary to model both spatial and spectral information. When considering spatial and spectral characteristics separately, it's evident that they can both be modelled as random processes (or sequences in the discrete case). There are however, major differences between the two underlying processes. Specifically, as previously stated, monochrome texture is stationary ergodic which means a limited amount of spatial texture can be used to reliably estimate a model from which an infinite spatial extent of texture can

be generated. In contrast, the spectral process is not stationary. Correlation across spectral bands is not relative, each pair may have a distinct dependency. Further, an image's spectra has finite bandwidth. Therefore, a realization of the spectral process is simply a 1D spectral curve. Obviously a model that attempts to abstract this information and use it to predict values at infinity is meaningless. What's needed is a method to generate spectral curves and then place them appropriately on a 2D grid.

4.1 Band Independence and the Global Color Space Transform

As previously mentioned, Heeger and Bergen [30] used a principle components analysis (PCA) transform, i.e. "whitening", in order to create independent color channels that could be synthesized separately. After performing the whitening transform, each band was synthesized using a single monochrome synthesis step and the resulting bands were back-projected. Although this approach was shown to be effective on a number of synthesis examples, the independence assumption made by Heeger and Bergen might be too strong.

There are basically two potential problems with this type of independence assumption. The first is the obvious one based on whether or not the data is truly Gaussian. In order for a PCA transform to create a set of new independent random variables, the underlying data must be Gaussian. If the data is not Gaussian, the effect of a PCA transform is to simply remove the linear dependency in the data. This is a common issue that arises when using a parametric model in practice. Basically, the problem is that the data may not truly fit the assumed parametric model.

However, a more profound problem is that multi-channel texture is basically a vector random sequence containing both a spatial and spectral component that are not always separable. In other words, the underlying distribution describing the joint distribution of radiance values does not factor into a product of two distributions; one spectral and the other spatial.

This lack of separability severely limits the effectiveness of the Heeger and Bergen approach. Their independence argument suggests that it is possible to correctly synthesize multi-channel texture by linearly combining the pixels in a set of uncorrelated bands. The only stipulation they place on these bands is that each must display the appropriate spatial characteristics vis-a-vis the

whitened sample texture.

To clarify this stipulation consider that the bands in the whitened sample texture are each still a texture. Each of these textures can be described by a different random process, which in turn can be used to generate a corresponding synthetic band. According to Heeger and Bergen's model, as long as each synthetic band appears to be generated by the corresponding random process, it should be possible to combine them through a back-projection or *coloring* operation to synthesize the multi-channel texture. In other words, as long each individual band is synthesized "correctly", then the resulting back-projected multi-channel texture will be "correct". Unfortunately, this is not always a valid assumption.

In order to test this theory, consider the following experiment. Starting with an RGB texture sample, perform a whitening operation on the bands. The result is three decorrelated bands. By simple back-transformation, the original texture is produced. In some sense this could represent an ideal case of the Heeger and Bergen model since the decorrelated bands are identically "correct" and by definition will produce the "correct" texture upon back transformation.

Now consider what happens if one of more of the decorrelated bands are shifted relative to each other before back transforming. Does the resulting texture appear valid? The short answer is not in all cases. What's more, it doesn't matter if the data is Gaussian or not. The results of this experiment are shown in Figure 15. In this figure, the image on the right is transformed into PCA space. Then a single band is shifted and the result is back transformed.

The results show that there is almost no detectable change after shifting a band. This is however, misleading. In the original texture the principle component dominates the residual color bands. This means most of the texture information is coming from one band. If the eigenvalues are rotated to equally represent each band, the effects of a shift are very noticeable Figure 16.

To reiterate, under Heeger and Bergen's model, as long as each decorrelated band displays appropriate spatial characteristics, then they can be combined through a coloring transformation to produce the correct texture. Clearly shifting a band does nothing to affect its spatial characteristics. After all, each decorrelated band is still a texture that is assumed to be stationary, meaning spatial correlations are *shift-invariant.*

Now consider the purpose of the independence assumption used by Heeger

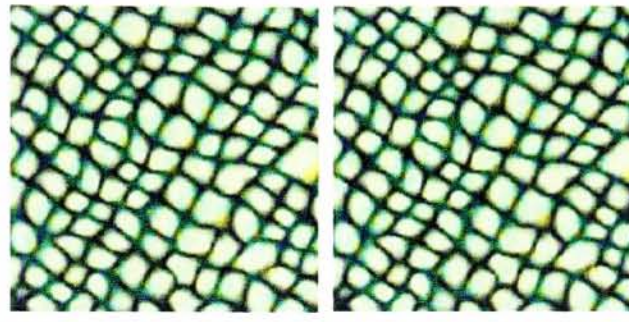


Figure 15: First experiment shifting decorrelated bands.

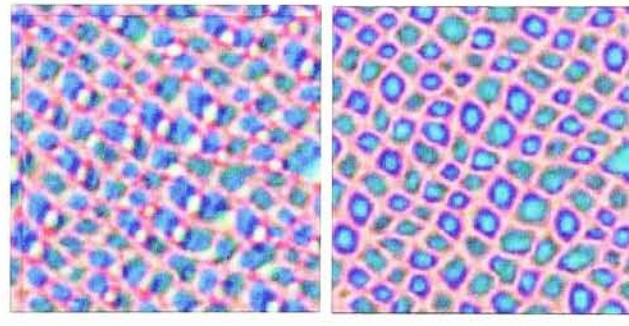
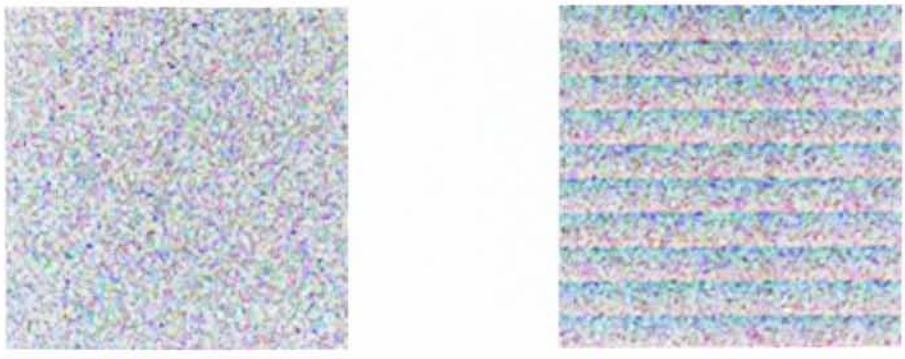


Figure 16: Second experiment shifting uncorrelated bands.

and Bergen. Basically, without the independence assumption they need to consider cross-band spatial dependencies, i.e. cross-correlation, that in turn make the problem more complex. In other words, without the independence assumptions, it becomes necessary to "line up" the bands with respect to each other. But the preceding experiment suggests that simple decorrelation of the color channels doesn't preclude the need to consider *cross-band spatial correlation.* As mentioned, the problem is that the global color space doesn't contain any spatial information. Provided that the joint spatial/spectral distribution is separable, this doesn't present any problem since in this case the cross-band spectral correlation doesn't depend on a spatial component.

However, consider another experiment where this is not the case. Repeat-



(a) (b)

Figure 17: Two images with the exact same pixel statistics.

edly sample a separable, zero mean, multi-variate Gaussian distribution with unit variance in each dimension. Now arrange the samples in a 2D pattern. The result is a multi-channel texture. Now, consider this the best-case scenario after whitening an otherwise unknown source texture. In other words, imagine successfully decorrelating a color space to create statistically independent channels. Remember that the random variables in a multi-variate Gaussian distribution are independent after a whitening transform. Next, it can be shown that even in this idealized case, the dependency between spectral correlation and cross-band spatial correlation cannot always be removed.

If these Gaussian samples are first distributed randomly across a 2D grid (Figure 17 (a)), it would be reasonable to expect that the spatial cross-correlation would be very low across each pair of bands Figure 18. In fact, since the spatial arrangement of spectra is entirely random, the joint spatial/spectral distribution is separable in this case.

The situation is much different if the spatial/spectral distribution is not separable. Consider a new arrangement of the same spectra. This time arrange the spectra in a quasi-periodic fashion over a 2D grid so that each region has similar spectral qualities. An easy way to accomplish this is to divide an image into sub-blocks. Then within each sub block, arrange the spectra in raster scan order based on energy Figure 17 (b). This time it would be reasonable to expect high spatial correlation across pairs of bands Figure 19.

Since both images have the exact same joint spectral distribution, there

is obviously some additional information that a global color space transform could never capture. Specifically, this information is the dependency between spectral correlation and cross-band spatial correlation when the joint distribution is not separable. What's more, it doesn't matter what the underlying joint spectral distribution is. This last point should be fairly obvious if one considers the same experiment could be performed using virtually any underlying spectral distribution provided it is not degenerate. In other words, the experiment will produce the same results as long as the underlying distribution is sufficiently random so that the random dispersal of spectral samples on the 2D grid doesn't display high cross-band spatial correlations already.

In summary, the failure of a global space transform to factor the joint spatial/spectral distribution into a product of independent marginals, seriously undermines the effectiveness of the Heeger and Bergen approach. Of course this doesn't mean the approach needs to be abandoned totally. Rather it suggests that there may be more appropriate models that don't make such limiting simplifying assumptions.

Figure 18: Autocorrelation functions of two bands showing arbitrary correlation.

Liang et. al. [34] recognized this problem and included their own analysis suggesting that the Heeger and Bergen approach was insufficient for creating independent color channels. There work focused on the fact that certain features like edges and lines had responses in all three color channels. The effect was that cross-band correlation had a local spatial component. Liang et. al. introduced a new type of color space transform based on indepen-

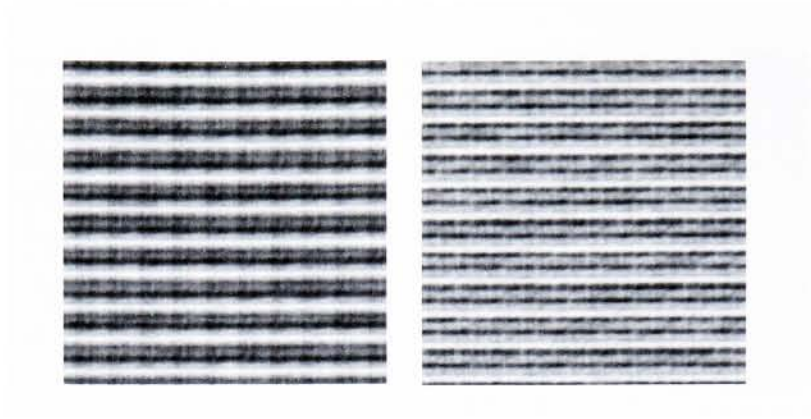


Figure 19: Autocorrelation functions in same two bands from image on left of Figure 17 (b) showing high spatial correlations.

dent components analysis (ICA). An ICA transform is used to minimize the mutual information over a set of random variables when the underlying joint distribution is not Gaussian. Although there are many ways to perform an ICA transform they all rely on some type of higher order statistic (HOS) to determine components of the data that are correlated. Remember that only the first and second order moments in a Gaussian distribution are unique. Therefore, if a distribution is not Gaussian there must be dependency information in the higher order moments. By using a combination of independent components analysis and a Gaussian scale mixture model (GSM) they were able to effectively cancel the local cross-band variance scaling resulting in nearly independent color channels.

Since local spatial information was included in their normalization scheme, Liang et. al. was able to separate spatial and spectral processing. After rendering the color channels as independent as possible, a single monochrome texture was synthesized in the dominant luminance channel, i.e. the channel containing the most spatial information. The remaining channels were determined by modulated the synthesized texture with variance scaled noise. A final back-projection into the original color space resulted in the final color texture. Results using this technique were impressive, but may not generalize to many bands.

The problem is that while a PCA or ICA transform are both invertible, the use of the GSM model can result in an overall loss of information. This

information loss over three bands would likely be exaggerated when extended to higher spectral dimensions. The result is that the spectral characteristics of the input and synthetic texture are not rigidly coupled. It would be desirable to ensure that the color space distributions of a sample and synthetic texture sample match as closely as possible. One step in this direction is to limit the use of spectral data to that found in the sample texture.

This is the how Wie and Lei [47] extended their resampling work to color texture synthesis. Rather than generate gray scale texture pixel-by-pixel via heuristic search, they copied individual spectra at each synthesis step. Clearly the conditioning of spectral information based on a local context neighborhood is a step in the right direction. As was previously stated though, the pixel-by-pixel search method is too slow for a practical application, and generally requires human input, something that should be avoided. What's needed is a robust/efficient way to generalize this concept to high spectral dimensions. Like the approach taken by Liang et. al. [34], it would be nice to synthesize texture spatially in a single band. Anything more would run the danger of becoming computationally burdensome. This band then becomes a constraint on the placement of spectral information. By incorporating local spatial context, the hope is that it is possible to correctly place spectral curves drawn from a random process estimated by a sample set of spectral data.

4.2 Modelling Spatial/Spectral Information as a Generalized Random Field

Part of the success of monochromatic texture synthesis algorithms is that monochromatic texture can be modelled as a stationary 2D random field. As previously mentioned, this means that the information in the image is not dependent on global position. An additional assumption of ergodicity means that any sample of sufficient size carries all the information necessary to define the underlying process that created the texture. Put in practical terms, this suggests that all spatial correlations in a texture will repeat over the ensemble of similar texture images Figure 20. The practical significance of this fact is that texture contains redundancy. This means that information extracted from a relatively small source can be extrapolated ad infinitum.

In contrast, the spectral process is not stationary. To begin with, the spectral information has finite extent, implying a global dependency on cor-

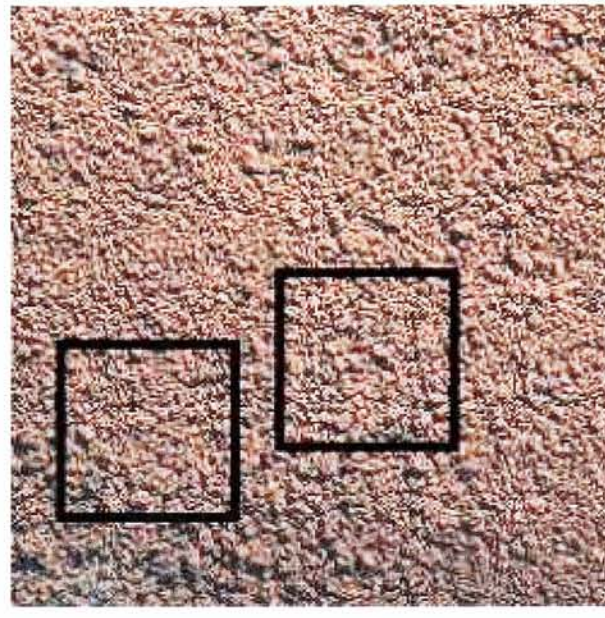


Figure 20: Texture can be modelled as a stationary ergodic random process.

relation. Therefore, the assumption that the correlation between bands is a function of relative distance is not valid. This implies one cannot generate novel spectral information using a technique tailored to stationary ergodic processes. To illustrate this point, consider how quilting is used to create texture. Entire regions of 2D texture are basically rearranged spatially while preserving the original characteristics of the texture. The equivalent operation in the spectral domain would be to rearrange band limited regions of the spectral information. This is obviously not valid or even meaningful.

Having established the fact that there is a fundamental difference between the statistical characteristics of the spatial and spectral information, a model must created that accounts for this difference. In monochrome texture synthesis the elemental building blocks are grey-level pixel values. As a generalization, one can describe multi-channel texture synthesis using spectral curves as elemental structures. In order to turn this description into a model, one needs a way to arrange spectral curves over a 2D grid so that the correct spatial properties are maintained in each band. In theory, one should

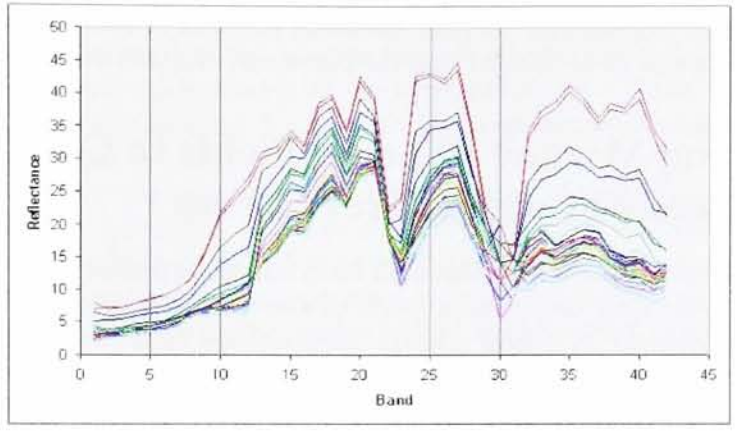


Figure 21: Two pairs of bands at equal spacing showing different correlations.

be able to synthesize a single spatial band of texture and then condition the choice of spectral curves based on the local context defined over the texture channel.

4.3 Claude Shannon and the Markov Assumption

Work by Claude Shannon in 1948 [42], suggested that realistic English sounding sentences could be generated by capturing local statistical dependencies drawn from a corpus of text. Using the Markov assumption it should be possible to condition the ith word in a sequence on the previous n words. By seeding a sentence with a randomly chosen word, a Markov process can generate sentences (albeit not necessarily semantically correct).

"One morning I shot an elephant in my arms and kissed him."

By analogy, instead of using words, a lexicon of spectral curves could be sampled where an example image cube forms the corpus from which the statistical characteristics of a Markov Random Field are drawn. Using this analogy it should be possible to consolidate the spectral information into elemental structures and rely on a single monochrome texture channel to condition the placement of spectral curves. What's needed is a reliable way

to synthesize monochrome texture, the result of which is used to estimate a Markov Random Field to condition the placement of spectral curves.

5 Using Markov Random Fields to Condition Placement of Spectral Curves

The first method presented in this work for synthesizing multispectral texture works by generating texture in a single channel and then selecting spectral curves conditioned on a local context neighborhood from that channel. In this sense, the resampling approach is generalized to identify spectral rather than spatial information. To illustrate this point, the task is no longer to grow the spatial extent of a synthetic texture using pixel-by-pixel resampling. Instead, the spatial extent of the synthetic multispectral texture is already defined after the single monochrome synthesis step in the principle band. Now the task is to use the same principles behind resampling to generate spectral information. Specifically, the task is to estimate a non-parametric Markov Random field (MRF) that describes the likelihood of observing a spectral curve given a particular spatial neighborhood. In this model, the Markov Random field has been generalized to allow the conditioning of a vector valued spectral curve, on a vector valued local neighborhood. This introduces the notion of a *Generalized Markov Random Field Model* that will play a central role for developing many of algorithms presented in this work.

In the same way that a non-parametric MRF can be used to describe the likelihood of observing a pixel given its local spatial neighborhood, this new model describes the likelihood of observing a spectral curve given its local spatial neighborhood. This fact is the key to developing a framework that can abstract the notion of a pixel to include vector quantities of arbitrarily high dimension. Operating within a non-parametric framework, the dimension of the spectral information is not a factor in the determination of a generalized MRF.

In theory, this dimension can be arbitrarily large without affecting the model. The reason can be illustrated quite simply in terms of pixel-by-pixel synthesis. To reiterate, pixel-by-pixel synthesis depends on a set of similar neighborhoods that are identified in the sample image via a template matching operation. The template is derived from the local neighborhood surrounding the pixel location that is to be generated. By sampling a single

neighborhood from this set, a central location is identified in the sample image. The pixel value at this location should be highly likely to occur given the neighborhood configuration in the synthetic image. Hence, the scalar pixel value at this location is copied to the corresponding location in the synthetic texture.

Now consider the case where the sample image is multi-channel. Then, without loss of generality, an entire spectral curve could be copied over to the synthetic texture instead of copying a single scalar pixel value from the sample image. Although this operation is intuitively correct its success is largely based on how well the single channel used to select spectral curves represents the spatial information in the entire cube. Naturally the choice of this single channel is critical.

Using the rich calculus of linear matrix algebra, it is possible to define a color space transform that finds a single channel that contains the maximum spatial information (in a least squares sense) from the entire image cube. This is done using an independent components transform (ICA), which is described in Appendix B and is similar to the Karhunen-Loeve transform (PCA). The idea is to decorrelate the individual bands which in turn identifies an orthogonal space scaled by the variances of the original signal when projected onto the new basis. The projection with the highest scaling contains the most variance. This band is referred to as the dominant luminance channel and is the band used to model the spatial correlation over the entire image cube.

5.1 Monochrome Synthesis Step

In this work a slightly modified version of Portilla and Simoncelli's complex analytic wavelet technique is used to synthesize the required principle band that is used to define the spatial model for the subsequent spectral generation step. As mentioned in the section describing the P/S algorithm, a single histogram modification replaces the individual low-order marginal constraints, i.e. mean, variance, skewness, and kurtosis. The algorithm is run for a fixed number of iterations, generally between 3 and 5. As a side note, empirical evidence suggests that a high number of iterations doesn't improve the convergence properties of algorithm.

5.2 Generating Spectral Information

After the principle texture band is synthesized using the Portilla and Simoncelli technique, another process is needed to condition the placement of the remaining spectral information. As mentioned, a search technique is used to find similar neighborhoods in the principle bands of the synthetic and input textures. The search is based on a Euclidean distance metric between two feature vectors extracted from a Laplacian decomposition of the sample and synthetic images. One vector is from the synthesis side and the other is from the analysis side Figure 8. This is essentially the same technique used by Debonet and Viola [13] and in particular Wei and Levoy [47]. Once a set of close matches are found between a location on the synthesis side and the entire texture band on the input side, the spectral information is copied by randomly sampling from the set.

Recall that under the resampling framework neighborhoods around central pixel locations define a spatial context. The key to capturing this context is to determine the distance or extent over which the context is non-redundant. Since texture often consists of textural elements of various scale, a mulitresolution approach is a natural representation. By representing all spatial dependencies at a given pixel location as a hierarchical parent structure, a multiresolution representation precludes having to choose or somehow induce an appropriately sized spatial neighborhood for any given texture type. This increases the chance that the algorithm will generalize well over a wide range of texture types. This fact was mentioned in the section outlining the Debonet and Viola method [13] and was described as one of the method's strengths. Unfortunately one of the major weaknesses of the method was that the coarsest scales must be tiled before the synthesis process can begin filling in the finer scales. This lead to the method's unfortunate bias towards structured textures. The situation is different now since the spatial extent of the synthetic texture has already been created. Now the Laplacian pyramids derived from the principle bands in the synthetic and sample textures are both unique and complete.

In this approach the principle bands in the sample and synthetic textures are expanded using $\lfloor \log(N) \rfloor$ levels, where N is taken from the sample (the assumption is that the synthetic is equal to or larger than the sample). This way the DC term (average value) at the top of the pyramid is discarded since it is redundant. The local context for a spatial location is defined as the parent structure obtained by forming a 3x3 neighborhood at each level

in the pyramid using circular boundary handling. In order to find similar neighborhoods the first step is to form the vector:

$$\vec{z}(x,y) = \langle N(l_0,x,y), N(l_1,x/2,y/2), N(l_2,x/2^2,y/2^2)..., N(l_k,x/2^k,y/2^k)\rangle$$

where N is a $3x3$ symmetric neighborhood at level l, x,y is the position in the original image, and k is the number of levels in the pyramid. The power of 2 scaling factor is needed because of the downsampling. Now each $\vec{z}(x,y)$, is matched against the set of vectors formed the same way from the sample pyramid. After finding the highest match, the residual spectral information is copied to the synthesis cube.

Since the generation of each spectral curve conditioned on the spatial context neighborhood is independent at each location, all the spectral information can be generated in parallel. This leads to significant reduction in time, which places the computational bottleneck on the single monochrome spatial synthesis step required to create the principle band in the synthetic image. The result is that multi-channel texture synthesis can be accomplished with time complexity independent of the number of bands given the appropriate hardware configuration. Beyond this natural parallel nature of the algorithm additional speed-up of the spectral generation step can be realized using optimized searching strategies. In this case, over the high dimensional feature space of hierarchical parent structures extracted from the Laplacian pyramid representations.

5.3 Nearest Neighbor Search Optimization

The *k-nearest neighbors* technique is an effective non-parametric classifier that shows good performance especially when used with large training sets. Unfortunately, for a single query point, the time complexity of the algorithm is on the order of nd where n is the number of training examples and d is dimensionality of the training set. This complexity arises from the fact that for a single query point, n distance calculations must be performed in a d dimensional feature space. To combat this computational burden, effective strategies have been proposed to reduce the number of distance calculations required to find the nearest neighbors of a query point.

Tree structured search strategies are very effective when the input dimensionality is relatively small, perhaps < 25. However, this efficiency is

known to degrade exponentially as dimensionality increases [41]. Codebook i.e. prototype strategies including hierarchical agglomerative clustering and vector quantization can be very effective for solving the *approximate nearest neighbor* (ANN) problem but suffer from the same sensitivity to dimension as tree structured methodologies.

The basic problem with tree structured searching is that it generally requires some type of geometric operation performed directly in the underlying feature space of the input patterns. An alternative strategy is to operate in a *metric space* rather than a *feature space*. This metric space is usually defined as the space induced by the $L2$ norm, provided it exists, and immediately allows us to use the triangle inequality to reduce the complexity of a nearest neighbors search.

The triangle inequality is powerful in the sense that it may provide insight during the evolution of a nearest neighbors search that may allow for elimination of many distance calculations. This insight comes in the form of lower-bounds on the distance to potential nearest neighbors. These lower-bounds can be used to eliminate distance calculations to points whose lower-bounds are known to exceed the current minimum distance. This is generally referred to as triangle inequality elimination (TIE). As will be shown, strategies for ensuring tight lower bounds can themselves be very sensitive to dimensionality of the training data.

An alternative strategy for minimizing the number of distance calculations in a nearest neighbor search is to simply reduce the number of training points. Often referred to as *editing*, *condensing*, or *Prototype selection* (PS). These techniques fall into two categories. The first category is *selection* which derives its name based on the fact that M prototypes are selected from a set of N training points. This is in contrast to *replacement* strategies that create a set of M prototypes by performing geometric transformations on the original training points. Examples of replacement strategies are hierarchical agglomerative clustering and vector quantization. As previously mentioned, these techniques are generally used to solve the approximate nearest neighbors problem which correctly implies that prototype selection strategies result in approximate nearest neighbor classifiers (ANN). This is in contrast to triangle inequality elimination which is an *exact* nearest neighbor classifier. Although prototype selection results in an approximation, the decrease in any classifier performance is hopefully overshadowed by the efficiency gains realized.

For the purposes of this work a simple strategy based on $KD-trees$ is

used to speed up the nearest neighbor searching process. The reason this algorithm is chosen is two-fold. First, efficient software is readily available. Second, KD-trees inherently provide an approximate solution to the nearest neighbor problem which is generally sufficient for this work. However, with simple extensions to the algorithm the solution can be made exact and in fact the entire search process is easily generalized to find neighbors within a user specified bounding ball. This is an particularly attractive aspect of the algorithm used in this work. Refer to Appendix D for a description of the KD-tree algorithm.

5.4 Pseudo-Code

In summary, a brief outline in psuedo-code illustrates the spectral generation technique:

```
N := Noise(Channels)

P/S-SpectralGen(Sample,N,iterations)
    S_ICA = ICA_Transform(Sample)
    P/S-TexGen(S_ICA(0),N(0),4,4,iterations)
    Lap_S = BuildLaplacian(S_ICA(0))
    Lap_N = BuildLaplacian(N(0))
    for x,y in N(0)
        z = CreateParentVector(Lap_N[x,y])
        <x',y'> = FinSimilar(z,Lap_S)
        CopySpectra(N[x,y],S_ICA[x',y'])

    N = BackTransform(N)
```

6 Multispectral Quilting

As mentioned in the background sections, recent developments in monochrome texture synthesis include methods that rearrange whole regions of input texture. Generally referred to as *quilting*, these techniques are faster and generally more stable than pixel-by-pixel resampling. Despite being a simplistic

graphics based approach, good synthesis results have been achieved by a number of researchers. The key to the method is that when performing pixel-by-pixel resampling, it is very likely that a pixel's immediate neighbors would be good candidates for the next synthesis step. Quilting saves time by simply carrying over a relatively compact set of neighbors at each synthesis step. A particularly attractive aspect of quilting is that it can be generalized to high spectral dimension texture synthesis. The idea is to simply carry along all the spectral information rather than a single pixel value.

When selecting the placement of texture blocks it is important to make sure the boundaries of neighboring blocks match relatively well. In order to ensure a good match, an overlap region can be compared across the sample image to determine candidate regions. The process of finding candidate regions is equivalent to finding similar context neighborhoods between the synthesis and analysis images. This is the same operation that was performed in the previous generalized P/S based technique in order to generate the residual spectral information and is a derivative of the simple resampling technique.

Under the quilting framework it is possible to combine the generation of the spectral and spatial information in one step. In fact generalizing quilting to handle texture with arbitrary spectral dimension is straightforward after the principle luminance channel is determined. In this model the quilting step is performed in an orthogonal color space on the principle luminance channel. At each synthesis step all the spectral information in the subblock of texture to be quilted can simply be carried over to the synthetic texture. This is obviously reminiscent of the previous generalized P/S based approach except the explicit monochrome synthesis step is subsumed into a larger synthesis process. This in effect accomplishes the same goal as before except the spatial and spectral information is determined simultaneously. This algorithm is particularly straightforward and produces very good quality results in a minimum amount of time. At this point all that is left is to describe some of the particular components of the quilting algorithm that are used to obtain high quality results in an efficient manner.

6.1 Minimum Boundary Error Cut

Clearly the texture blocks that are quilted together will likely not match exactly across their boundaries. Efros and Freeman [20] proposed a novel way to minimize the effects of this edge discontinuity. They used a min-cut

procedure to find the optimal boundary between blocks. The same approach is adopted here where the only difference is that the min-cut is propagated across all channels. A somewhat simpler approach is to use alpha-blending at the boundaries. The major drawback of this approach is that the blending may introduce anomalous spectral curves.

The boundary error is defined as the magnitude of the difference between overlapping pixels. Expressed as a cost function the boundary error is defined as:

$$C_{i,j} = |S_{i,j} - B_{i,j}|$$

where C is the cost, S is the synthesis region and B is the current block region.

Finding the minimum boundary error cut is equivalent to a single source, shortest path problem. In the single source shortest path problem the distance from a single start node to each node in a graph structure is determined. A particularly efficient means for finding shortest paths through a single source network is to use Dijkstra's algorithm. See Appendix C for a detailed explanation of Dijkstra's algorithm.

6.2 Nearest Neighbor Search Over Boundary Overlap

In order to synthesis texture, quilting places regions of texture in a raster scan order. The overlap is defined over a region $1/6$ the size of N, where N is a user specified block size. There are three types of overlap, 1) left side overlap across first raster scan line 2) top side overlap down the first column 3) left and top side overlap on all internal blocks. In order to find similar neighborhoods quickly, a convolution operation is performed in the Fourier domain. In order to normalize the convolution output, three normalization functions are defined for the three overlap types Figure 22. Using three masks identifying the active boundary over which the normalized length of the sample image is computed. This function is then used to calculate a cosine similarity measure between a block boundary and the sample image. This cosine similarity is used to determine a set of neighbors based on a user specified threshold. As with other resampling methods the set of similar neighbors represents a non-parametric Markov Random Field defining the likelihood of a texture block conditioned on its neighbors. A random sample

from this neighbor set is used as the next texture block in the synthetic image. As mentioned a minimum cut procedure is then used to find the optimal border between texture blocks.

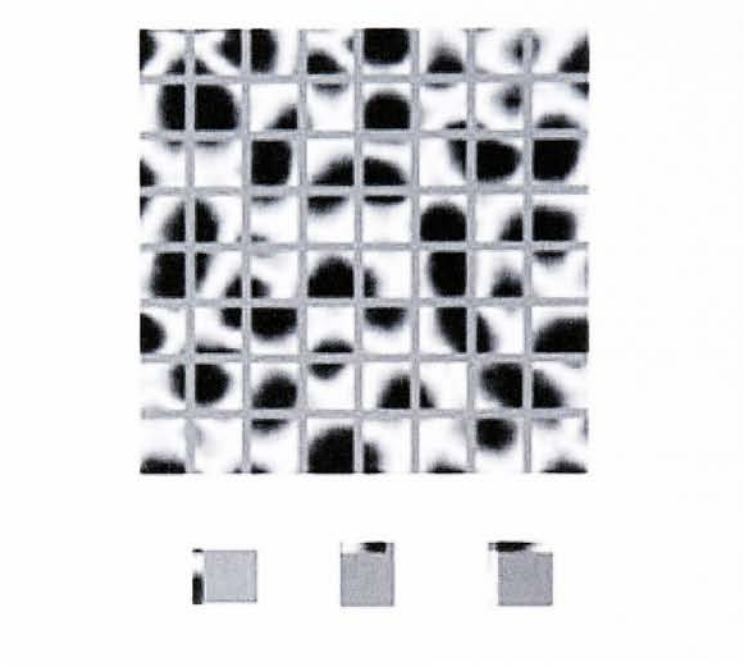


Figure 22: Quilting result making explicit the boundary overlap regions.

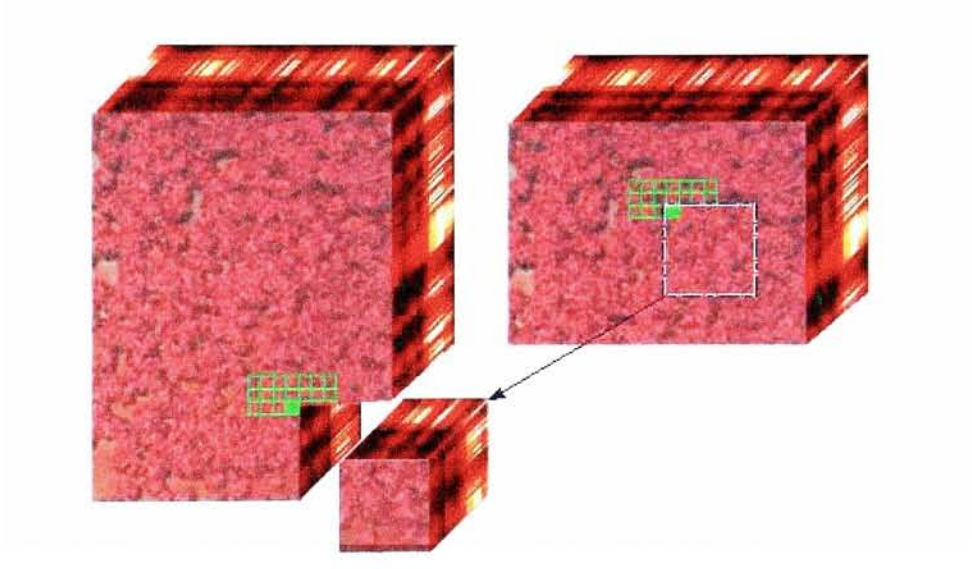


Figure 23: Boundary overlap determining current quilting step.

6.3 Pseudo-Code

In summary, a brief outline in psuedo-code illustrates the quilting monochrome texture synthesis technique.

```
N := Block_Size
Synth := Noise(Channels)

Quilt(Sample,N)
    Overlap = 1/6*N
    S_ICA = ICA_Transform(Sample)
    Initialize(Synth(0),S_ICA(0))
    Masks = Generate_Overlap_Masks()

    while x,y in Synth(0)
        if(y==0)        //top edge mask
            mask = Masks(0)
        else if(x==0)   //left edge mask
            mask = Masks(1)
        else            //internal region mask
            mask = Masks(2)

        Neighbors = FindSimilar(S_ICA,Synth(0)[x,y],mask)
        New_block = Random(Neighbors)
        Quilt_Block(Synth[x,y],New_block)
        x+=N
        if(x\%N != N )  //end of image
            y+=N
```

7 Results

A number of experiments are performed that demonstrate the effectiveness of various texture synthesis algorithms. Results are reported for monochrome, color and multispectral synthesis. It is natural to report monochrome results since the multispectral algorithms described in this work require a monochrome synthesis step. The color synthesis results are important because color textures can be viewed as images, readily providing a qualitative

assessment of the effectiveness of the multispectral algorithms operating in low dimensions. For both monochrome and color synthesis, results are reported as simple input-output image pairs with the only quantitative results consisting of algorithm timings. When reporting the multispectral results, the focus necessarily shifts towards quantitative descriptors of synthesis quality.

It is generally a difficult problem to determine quantitatively how well a texture has been synthesized. As a result, most of the literature relies on simple visual appearance to evaluate synthesis performance. This type of visual test is certainly an important aspect of evaluating the effectiveness of a synthesis algorithm. Trying to evaluate synthesis quality with a set of quantitative descriptors can be difficult and in some cases misleading. Of course, a visual test becomes problematic when dealing with textures that have high spectral dimension. As a result, a unique approach using k-means clustering is used to provide a visual representation of texture with high spectral dimension through sub-space projection methods. In addition to using this visual comparison, a set of metrics is chosen to provide a first order quantitative approximation to each model's goodness-of-fit.

Test data has been chosen that exhibits a good diversity of texture characteristics including both structural and stochastic textures. In addition to a set of standard natural textures, actual multispectral imagery indicative of textures encountered in the simulation domain are used. A number of examples are provided for each of the monochrome, color and multispectral experiments. In each case, high quality results as well as failures are reported. By including a good cross-section of synthesis results it is hoped that the reader may gain an insight into any inherent strengths or limitations that the synthesis algorithms may exhibit.

7.1 Monochrome Texture Synthesis Results

Since each multispectral synthesis technique relies heavily on a quality monochrome synthesis step, a number of algorithms are tested on a set of standard examples from the Brodatz [8] collection of gray-scale textures. The algorithms tested are 1) quilting with alpha-blending 2) quilting with min-cut 3) quilting with block size of one, i.e. pixel-by-pixel resampling 4) Portilla and Simoncelli (P/S). In addition to the synthesis results over a standard test suite, simple timing comparisons are made in relation to their asymptotic time complexity.

The first evaluation criterion is a simple timing test on a standard image. Timing comparisons where obtained on a desktop Pentium system running at 750 mhz. A standard 128x128 texture patch was used by all algorithms to produce a 256x256 gray scale synthetic texture. The P/S method took 320 seconds while quilting took just 120, 138 and 280 seconds for alpha-blend, min-cut, and resampling respectively. Refer to Table 1 for timing breakdown, where N represents synthetic texture size, S is the sample size, and n is the block size (for quilting only).

Method	Synthesis Time	Time Complexity
Quilting Alpha-Blend	120 seconds	O((N/n)(S)log(S))
Quilting Min-Cut	138 seconds	O((N/n)(S)log(S))
Resampling	280 seconds	O(NSlog(S))
Portilla/Simoncelli	320 seconds	O(iterations*(N)log($N^{1/2}$))

Table 1: Table showing timings and asymptotic time complexities of each algorithm.

Each monochrome texture synthesis algorithm has unique attributes. Quilting does a very good job of handling moderately structured textures as well as textures displaying multi-scale features Figure 24. Highly structured textures pose a slight problem for the algorithm mainly because synthesis quality is heavily dependent on neighborhood size Figure 25. The P/S technique shows reasonable results for a wide range of textures but can't quite match the spatial fidelity of quilting in some cases Figures 30, 31. The P/S technique has a tendency to introduce some artifacts and discontinuities and seems to have problems resolving multi-scale textural elements. This is somewhat surprising because it uses a multi-scale approach. At times edge discontinuities are apparent when using the quilting technique and it is still possible for quilting to get stuck in one area of the search space leading to instability of the algorithm. This is most evident in the pixel-by-pixel resampling method. On the positive side, resampling doesn't introduce the kind of edge artifacts that arise in quilting. The output results over a standard test suite of monochrome texture are presented in Figures 24-31.

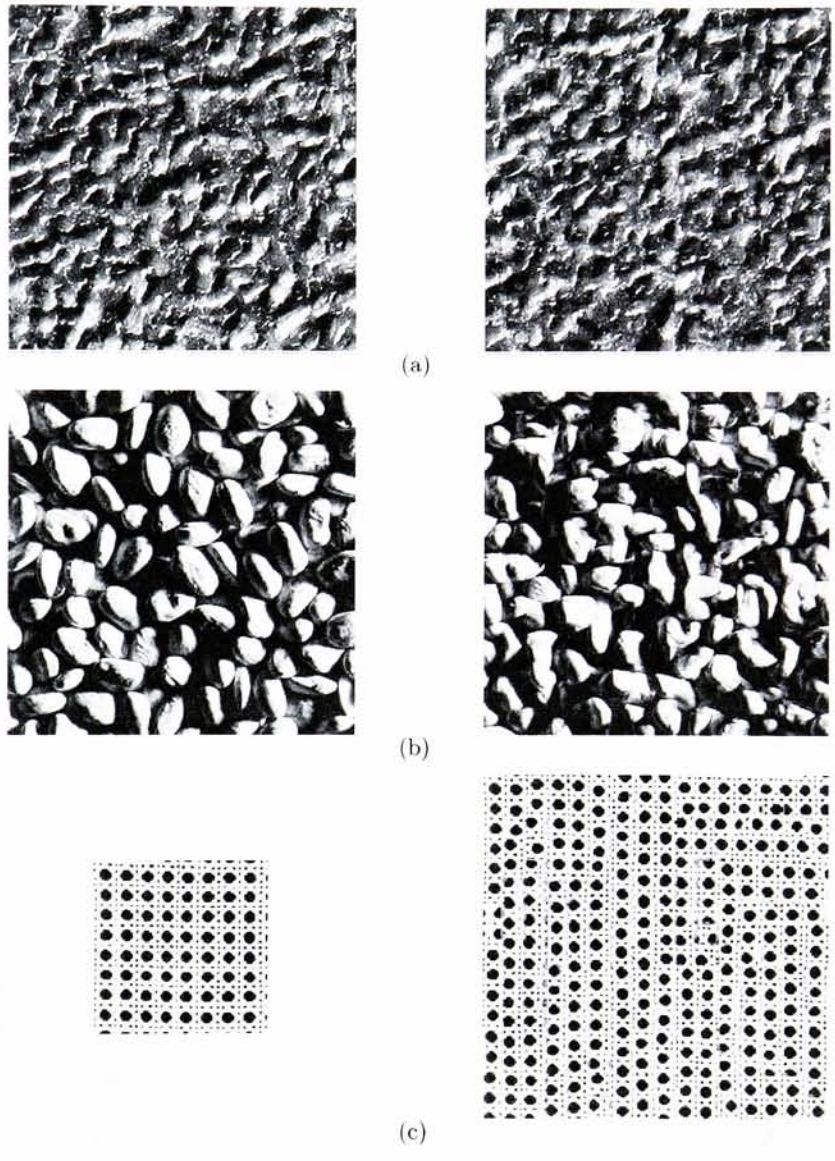


Figure 24: Quilting with alpha-blend using neighborhood size of 16.

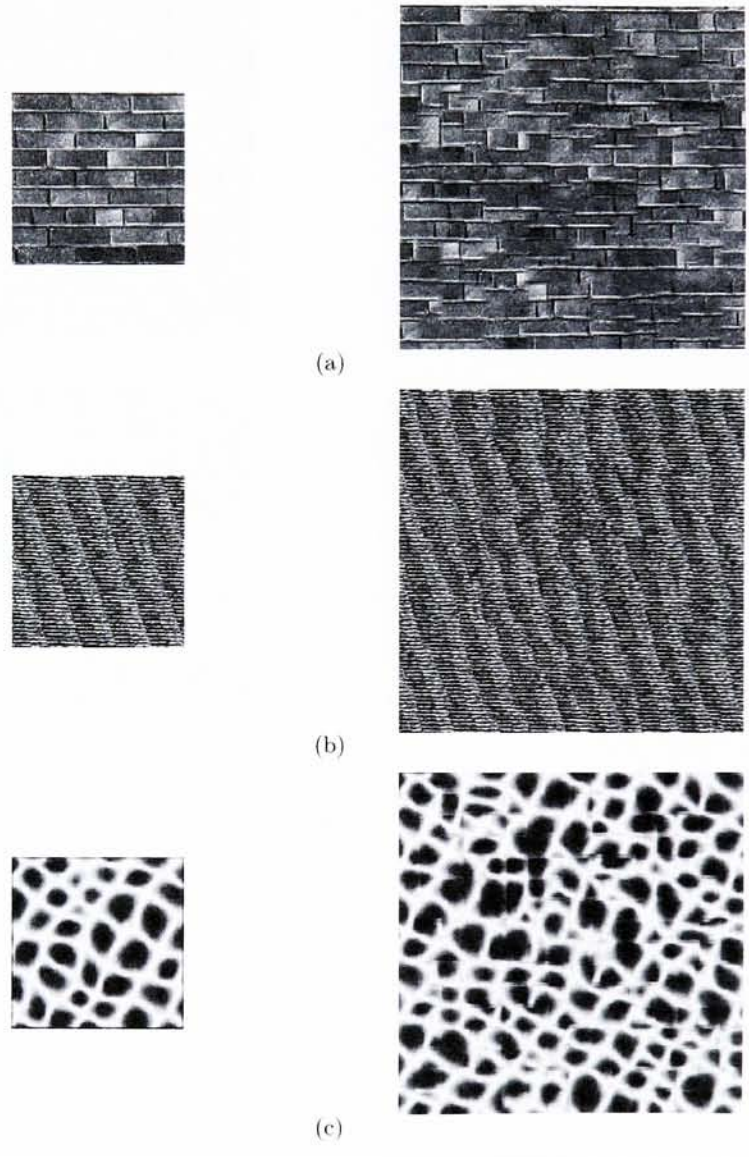


Figure 25: Quilting with alpha-blend using neighborhood size of 16.

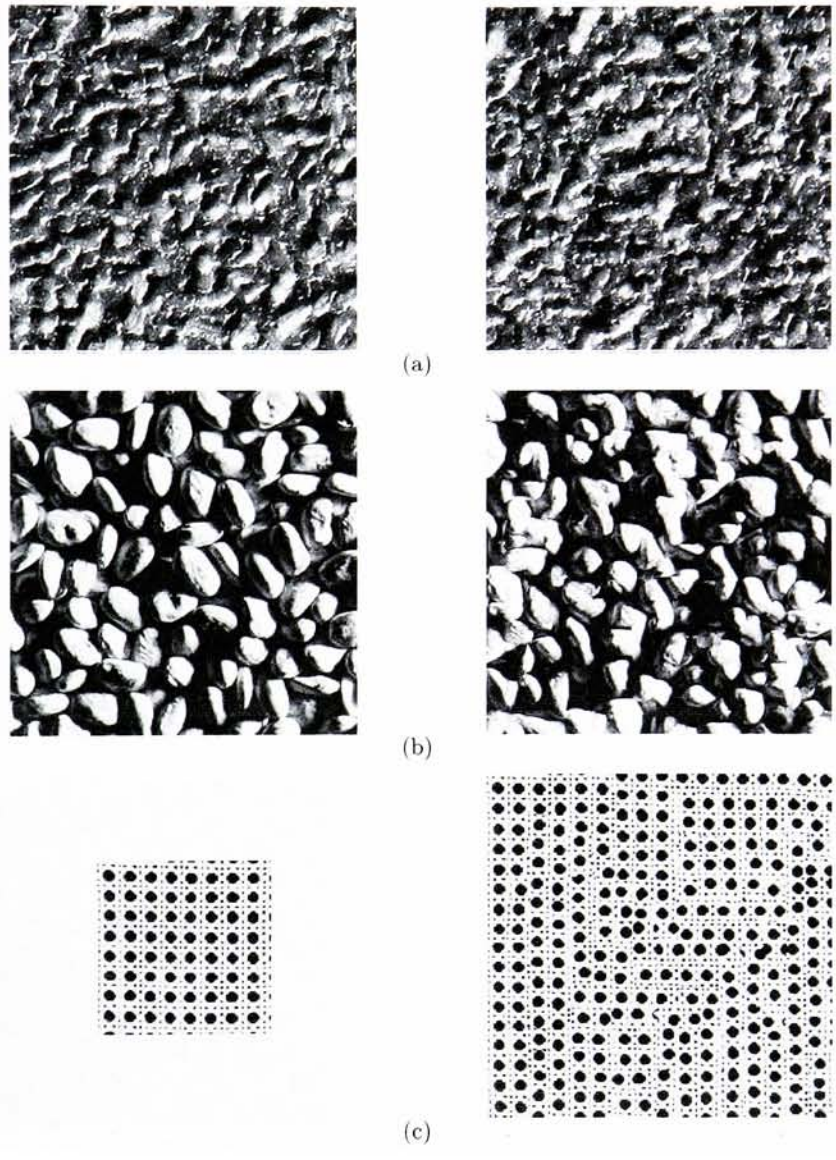


Figure 26: Quilting with min-cut using neighborhood size of 16.

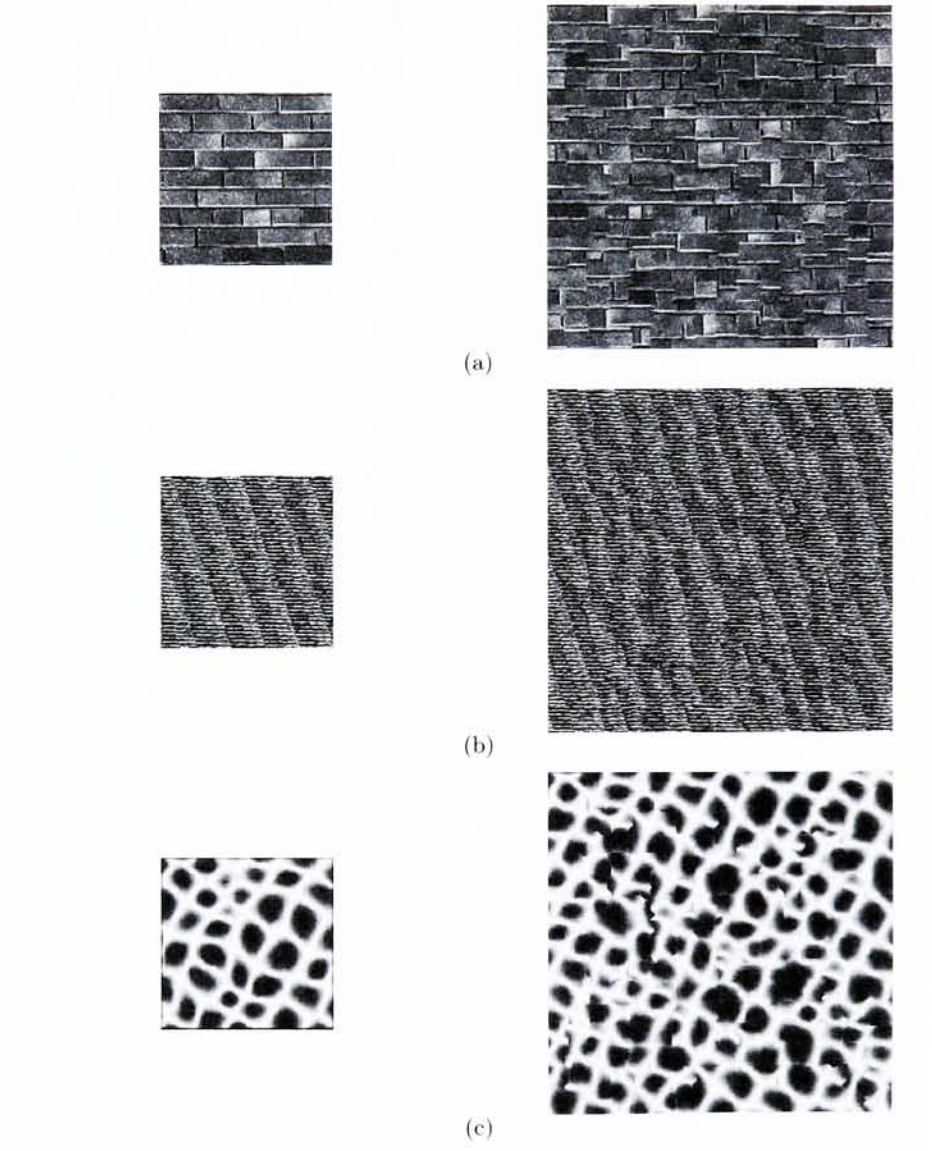


Figure 27: Quilting with min-cut using neighborhood size of 16.

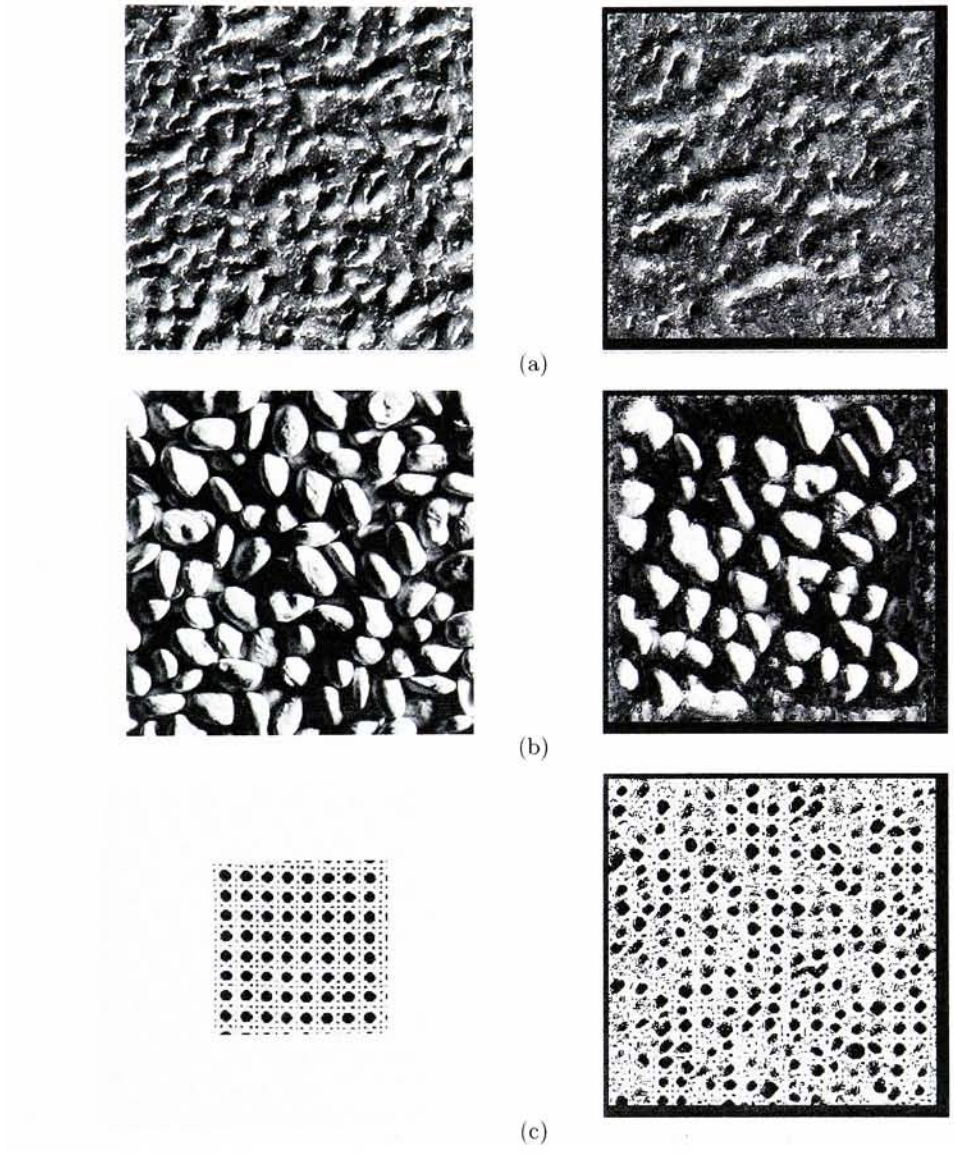


Figure 28: Pixel-by-pixel resampling using neighborhood size of 16.

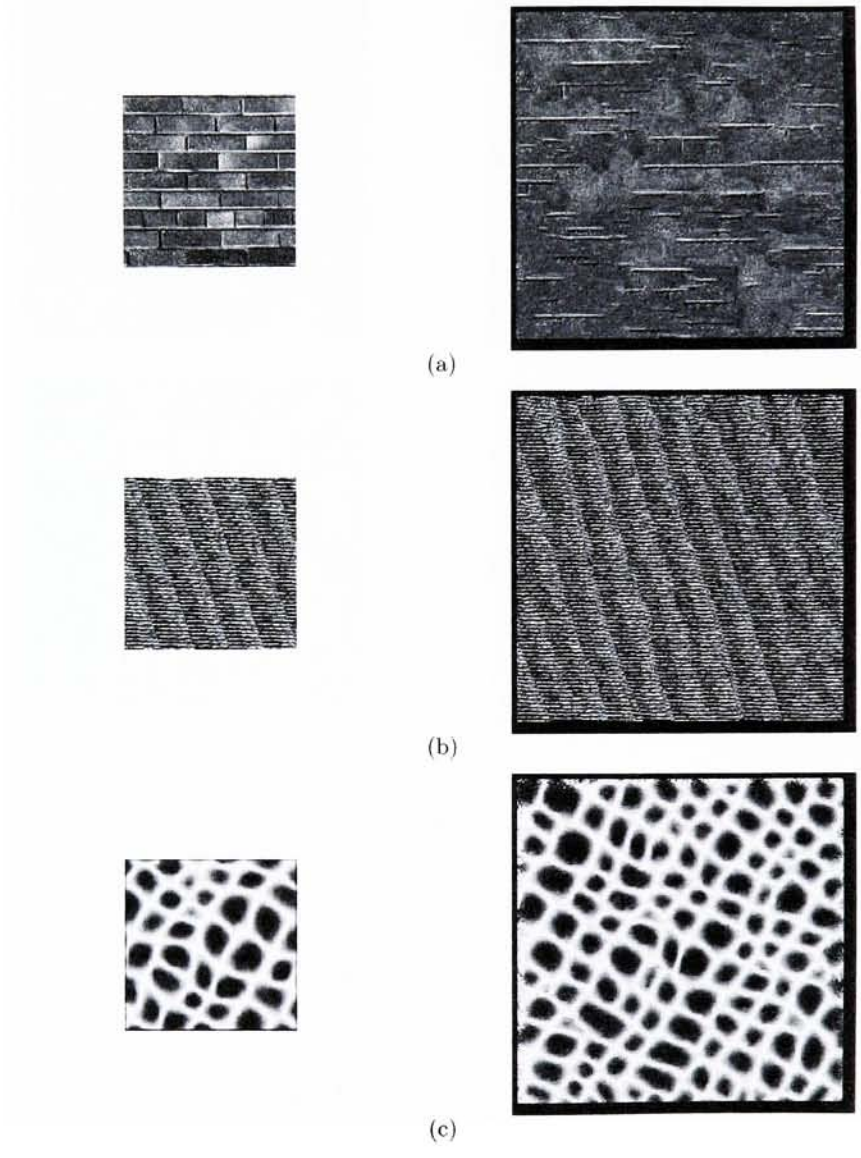


Figure 29: Pixel-by-pixel resampling using neighborhood size of 16.

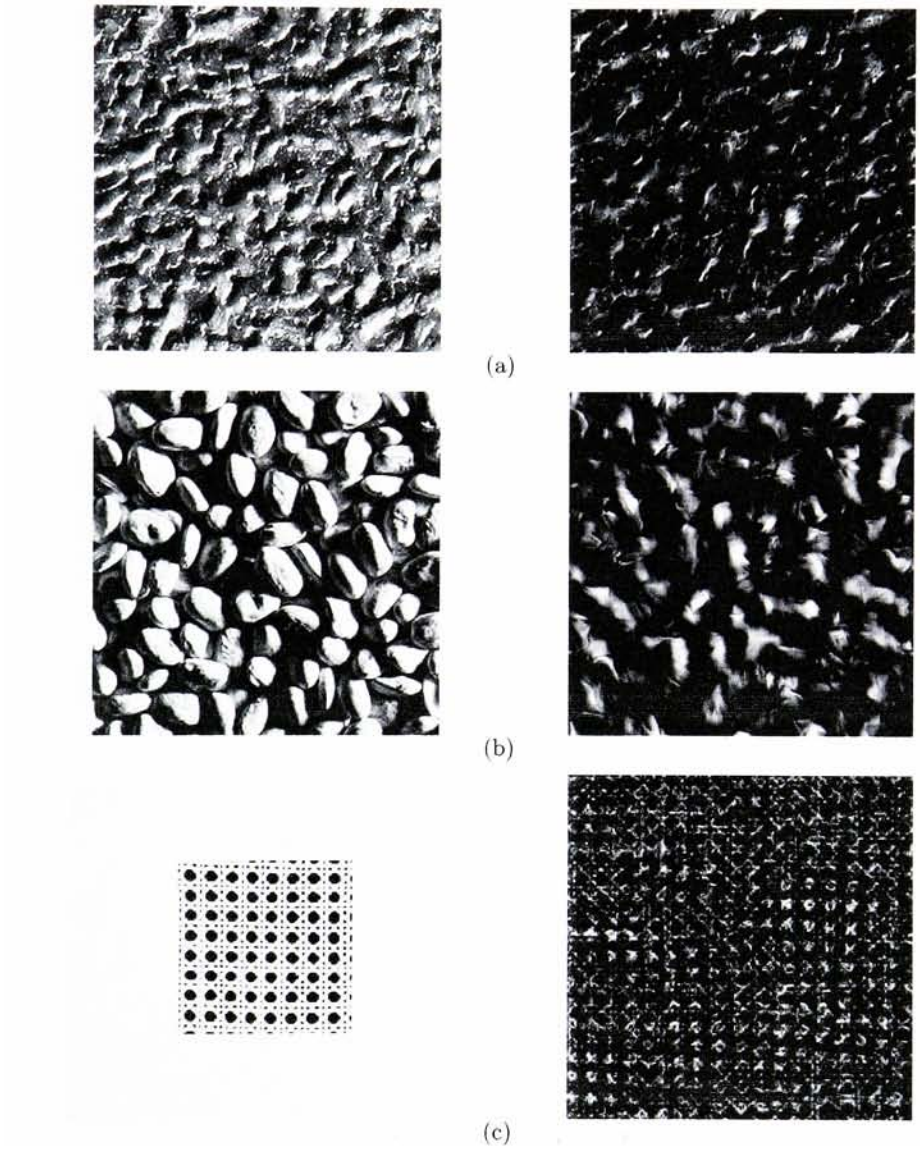


Figure 30: Synthesis results using P/S method.

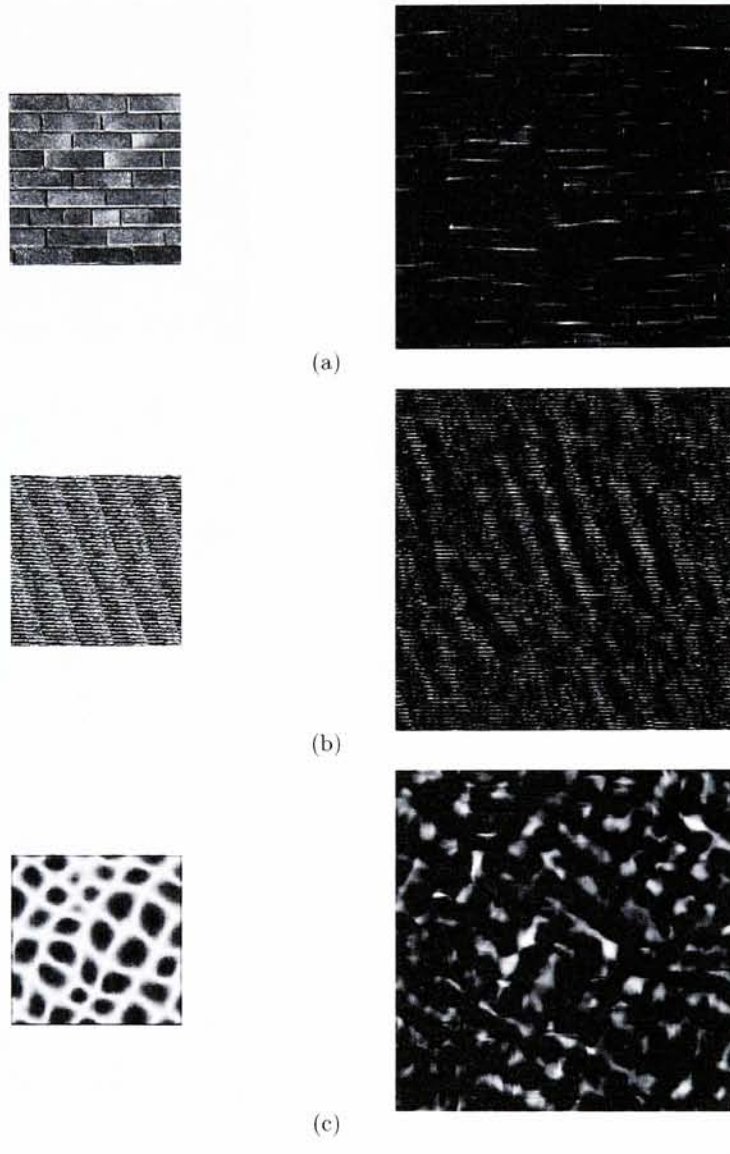


Figure 31: Synthesis results using P/S method.

7.2 Color Texture Synthesis Results

A number of examples of color texture are synthesized as both a comparison with the monochrome synthesis results and as an intermediate validation of the multispectral synthesis algorithms. The color texture images are diverse and include textures from aerial photographs of actual terrain as well as some natural and organic color textures from the VisTex collection [46]. Before presenting the synthesis results a simple timing comparison is again performed.

As the number of spectral channels increases from one to three, the relative speed of the various algorithms begin to diverge. As compared to their monochrome variants, the P/S color texture synthesis timings become much more disparate. This is because a parallel implementation of the resampling stage is not used. As a result, synthesis operations on a 128x128 input to 256x256 output now takes up to for times longer for the P/S method as compared to the quilting variants which remain virtually unchanged. Refer to Table 2 for timing breakdown where, N represents synthetic texture size, S is the sample size, and n is the block size (for quilting only).

Synthesis quality is general comparable to that seen in the monochrome case. This is to be expected since all spatial information is derived from a single monochrome synthesis step. In the spectral domain, color is reproduced well in most cases with the exception of the P/S method. In some cases the results are extremely poor Figure 43(c). It seems as though the method cannot achieve an appropriate spectral resolution. At times the color in the image becomes highly quantized almost binary in some cases, heavily skewed towards what appears to be an average value. This is most likely due to poor scaling of the marginal statistics in the principle band that in turn tends to dominate the resulting spectra. Color texture synthesis results for each algorithm are presented in Figures 32-43.

Method	Synthesis Time	Time Complexity
Quilting Alpha-Blend	141 seconds	O((N/n)(S)log(S))
Quilting Min-Cut	156 seconds	O((N/n)(S)log(S))
Resampling	310 seconds	O(NSlog(S))
Portilla/Simoncelli	1719 seconds	O(iterations*(N)log($N^{1/2}$))

Table 2: Table showing timings and asymptotic time complexities of each spectral synthesis algorithm.

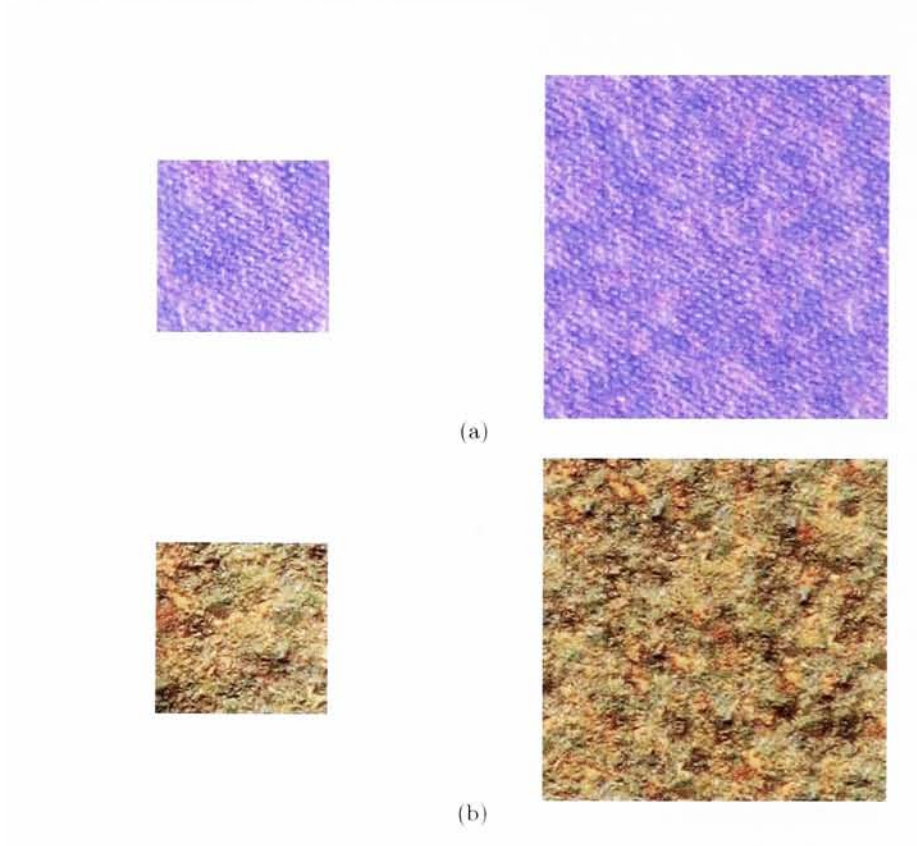


Figure 32: Color synthesis results, quilting with alpha-blend using neighborhood size of 16.

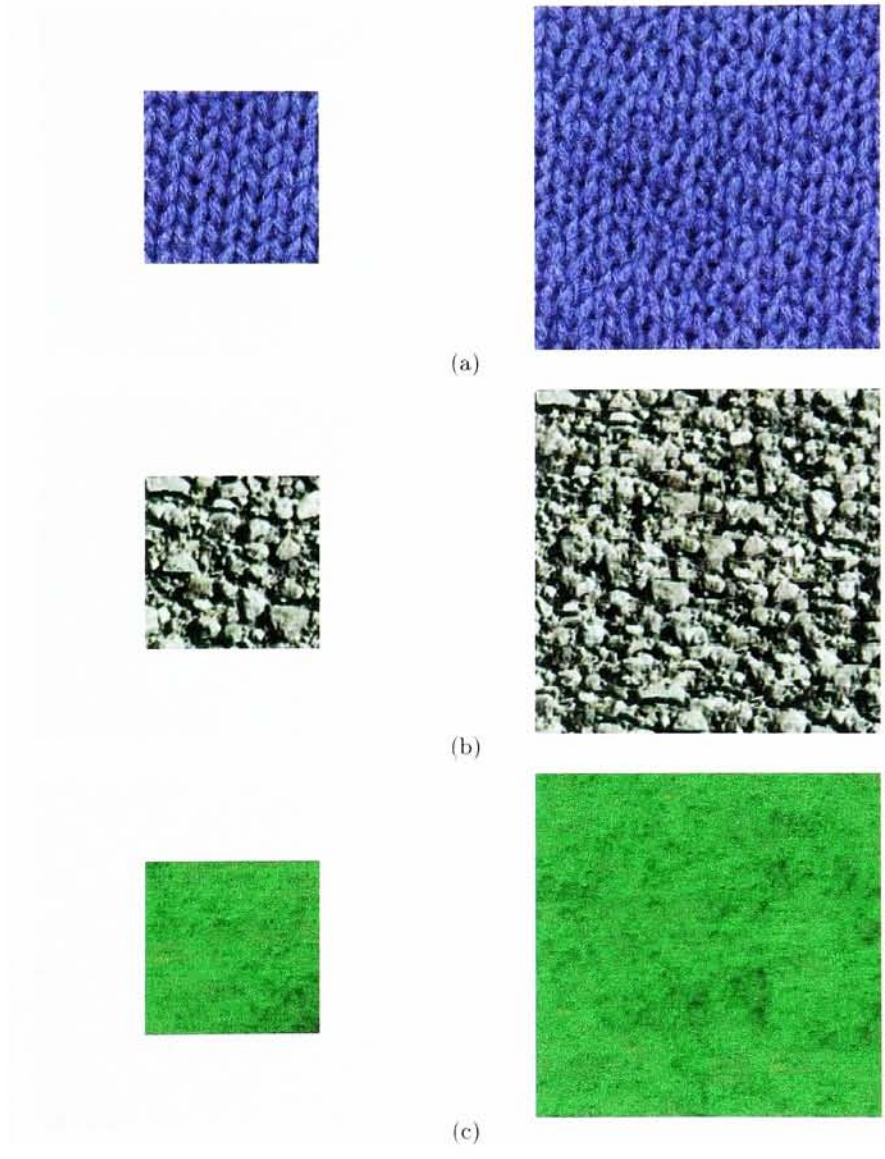


Figure 33: Color synthesis results, quilting with alpha-blend using neighborhood size of 16.

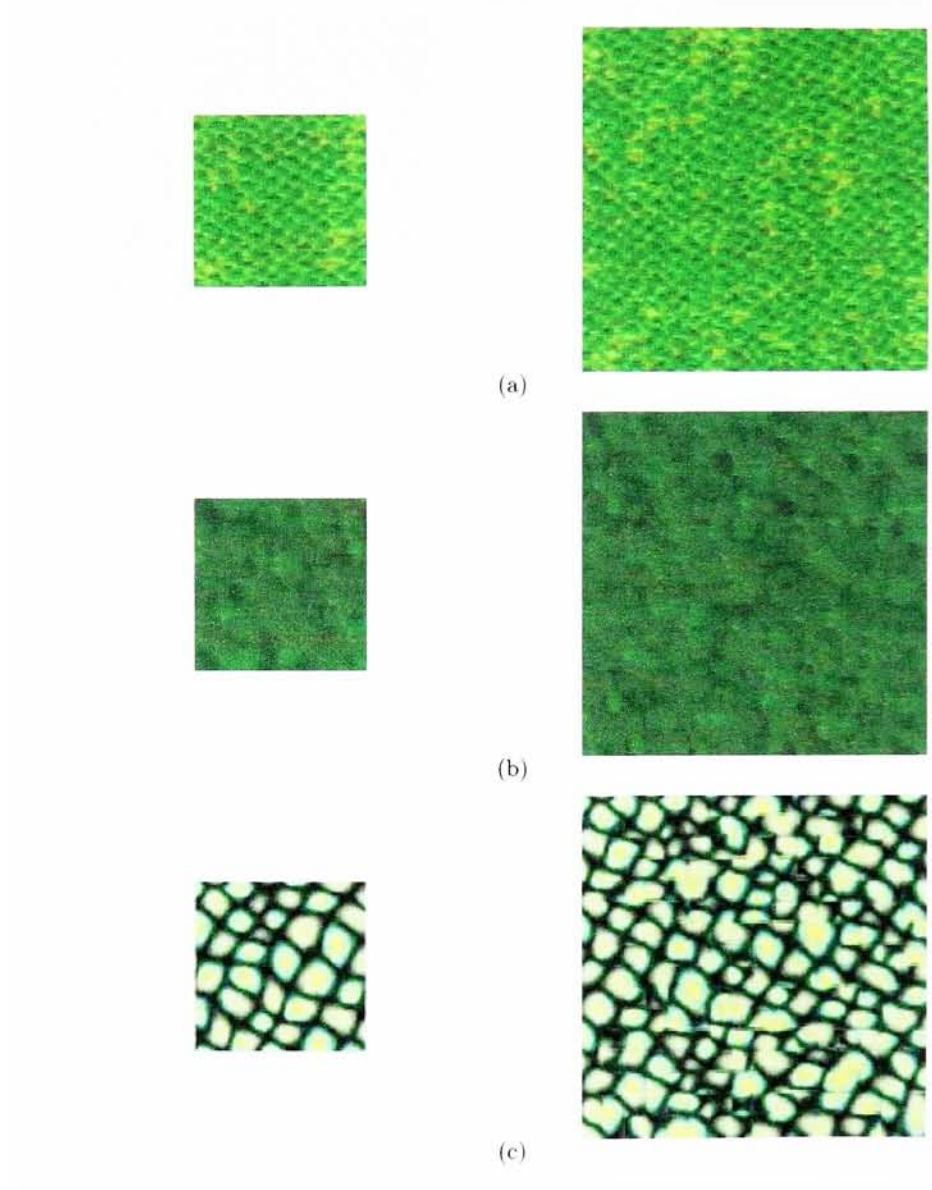


Figure 34: Color synthesis results, quilting with alpha-blend using neighborhood size of 16.

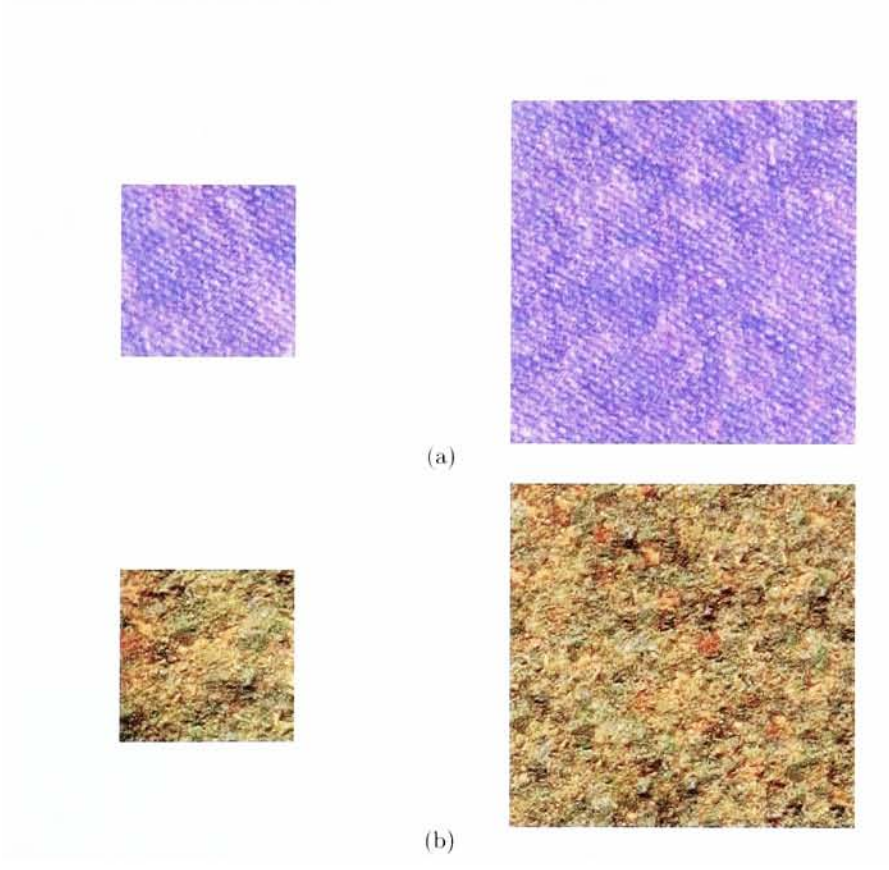


Figure 35: Color synthesis results, quilting with min-cut using neighborhood size of 16.

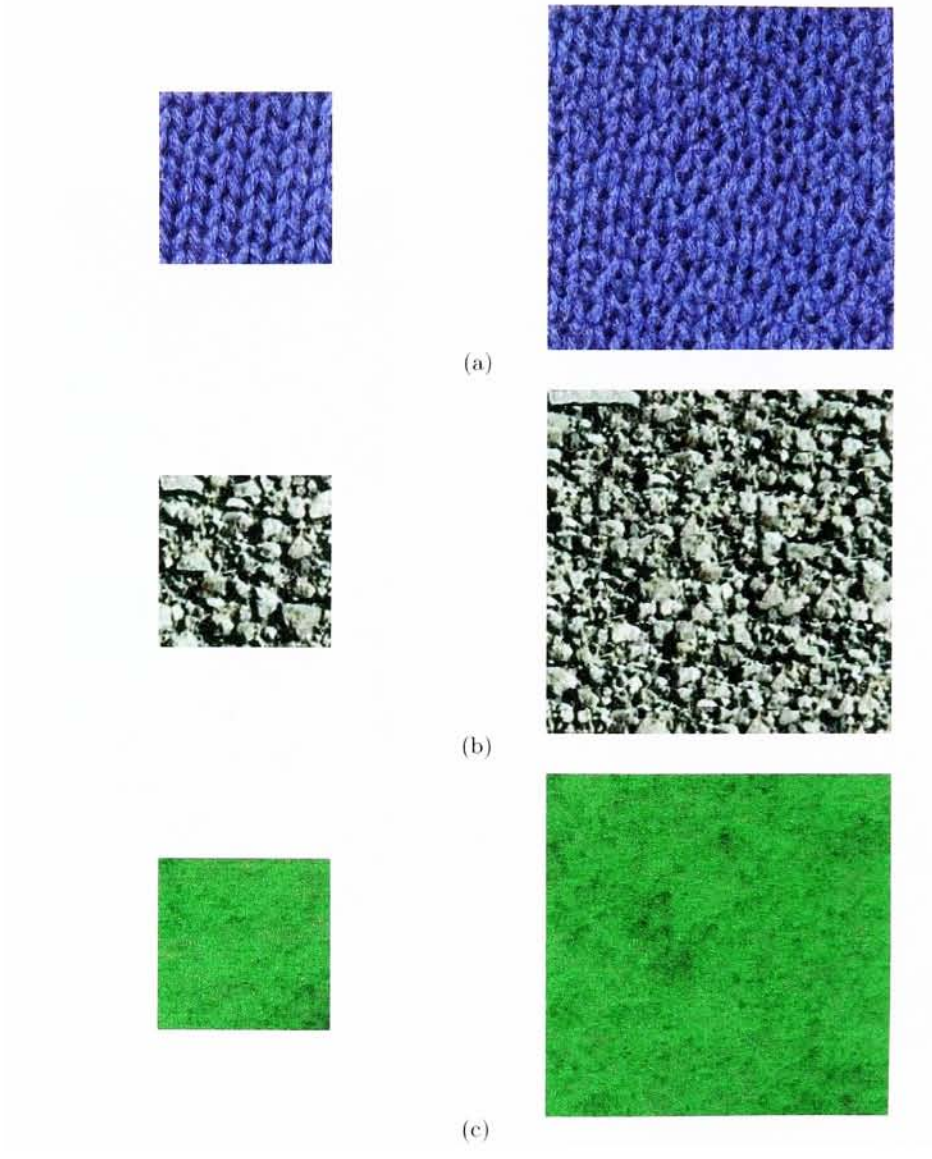


Figure 36: Color synthesis results, quilting with min-cut using neighborhood size of 16.

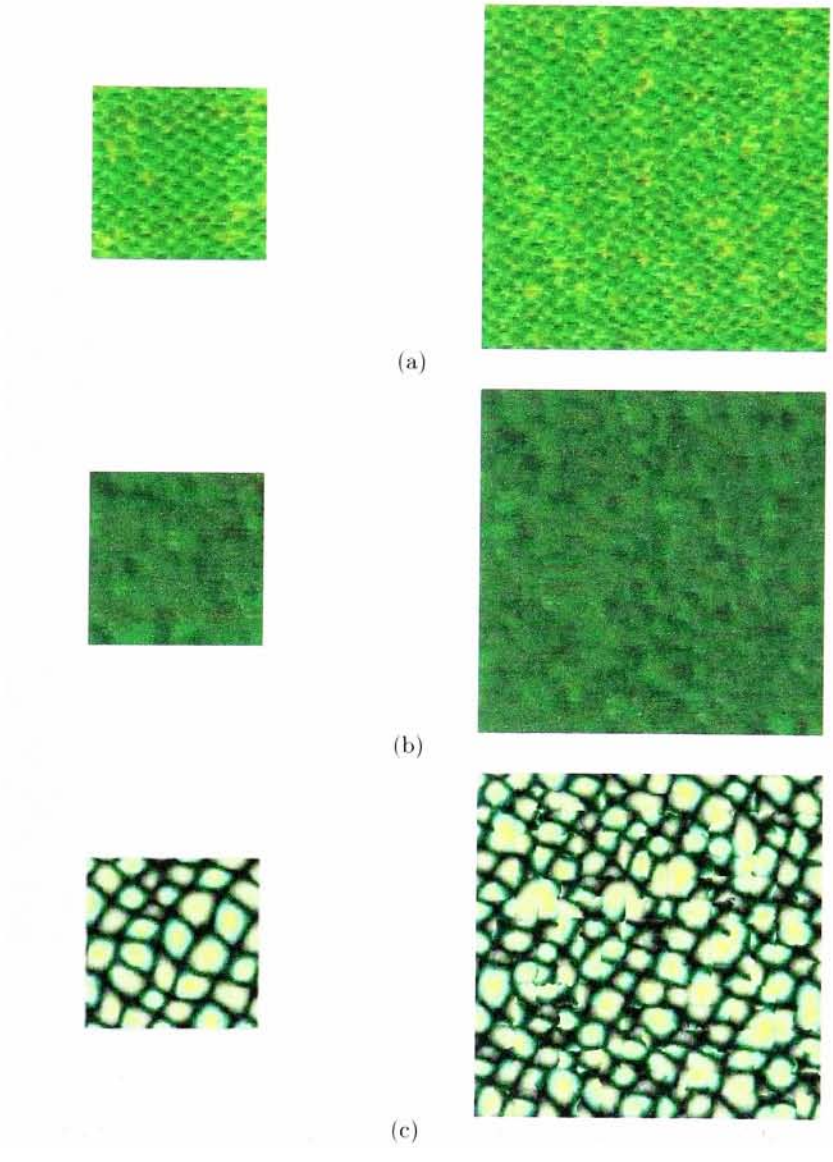


Figure 37: Color synthesis results, quilting with min-cut using neighborhood size of 16.

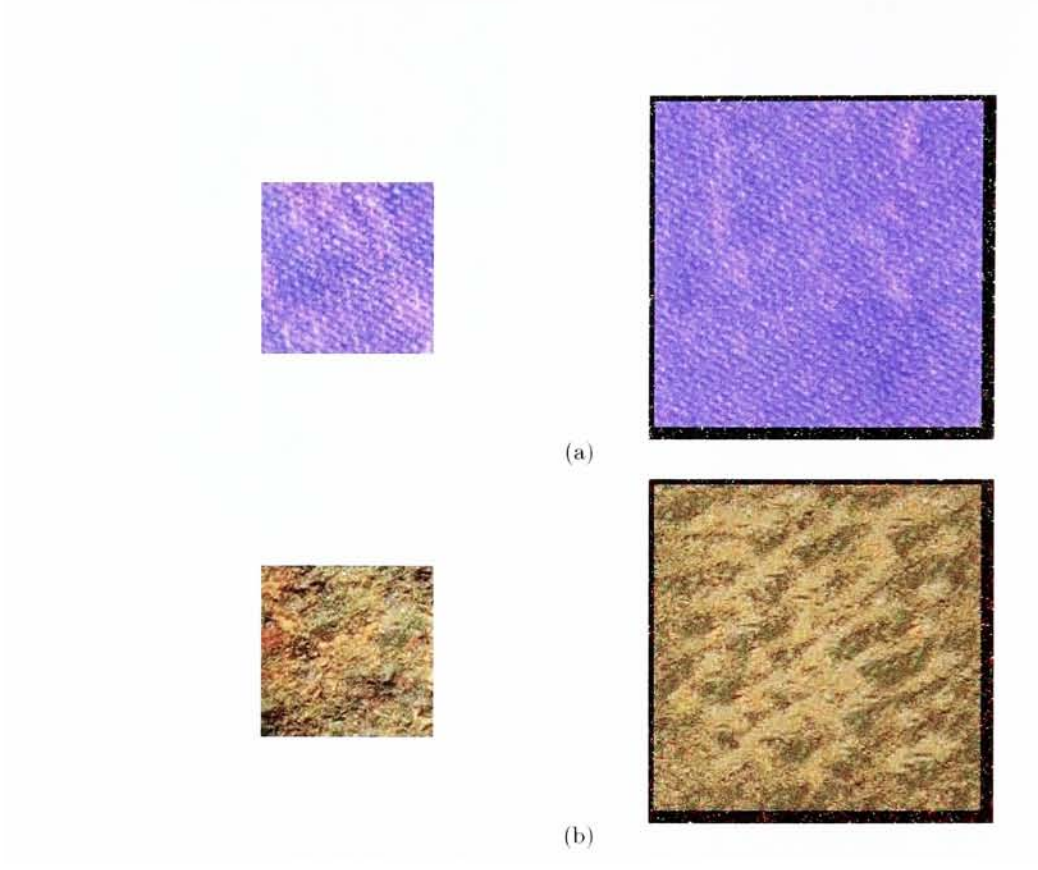


Figure 38: Color synthesis results, resampling using neighborhood size of 16.

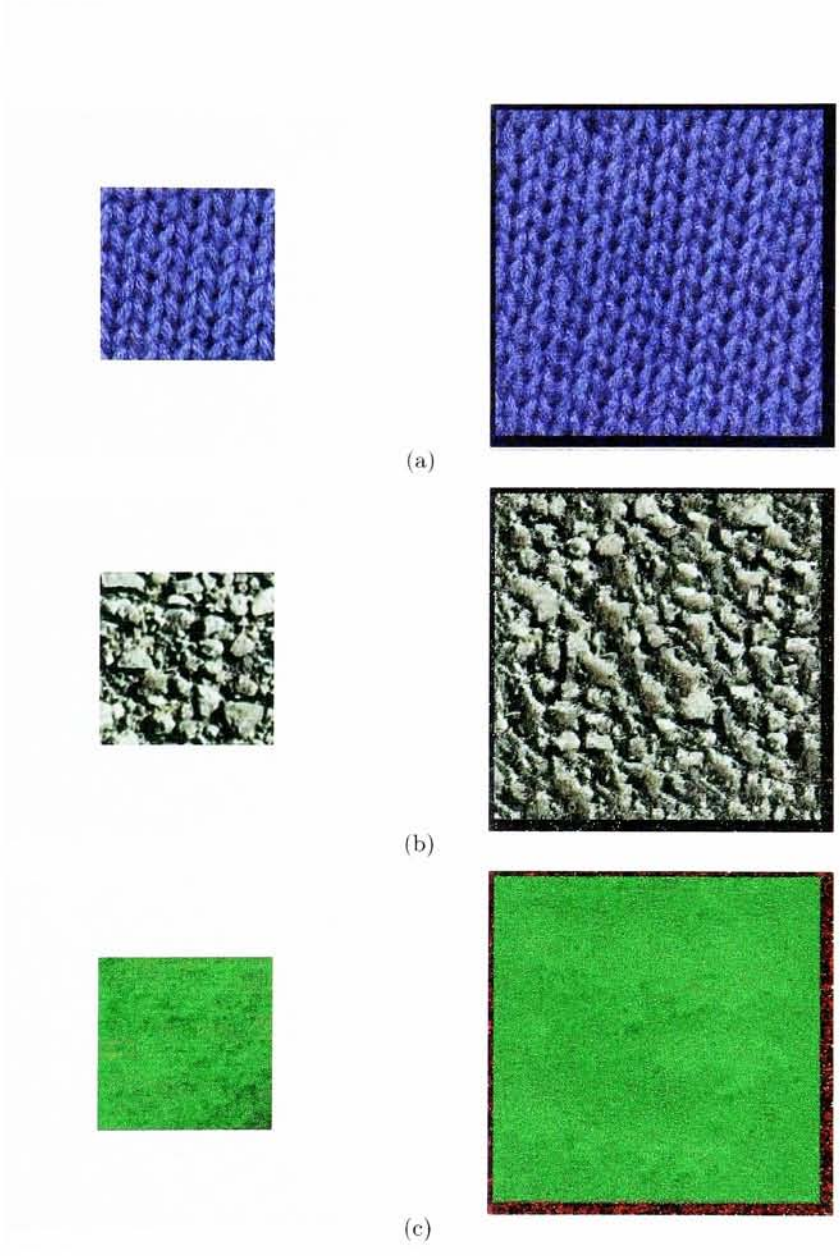


Figure 39: Color synthesis results, resampling using neighborhood size of 16.

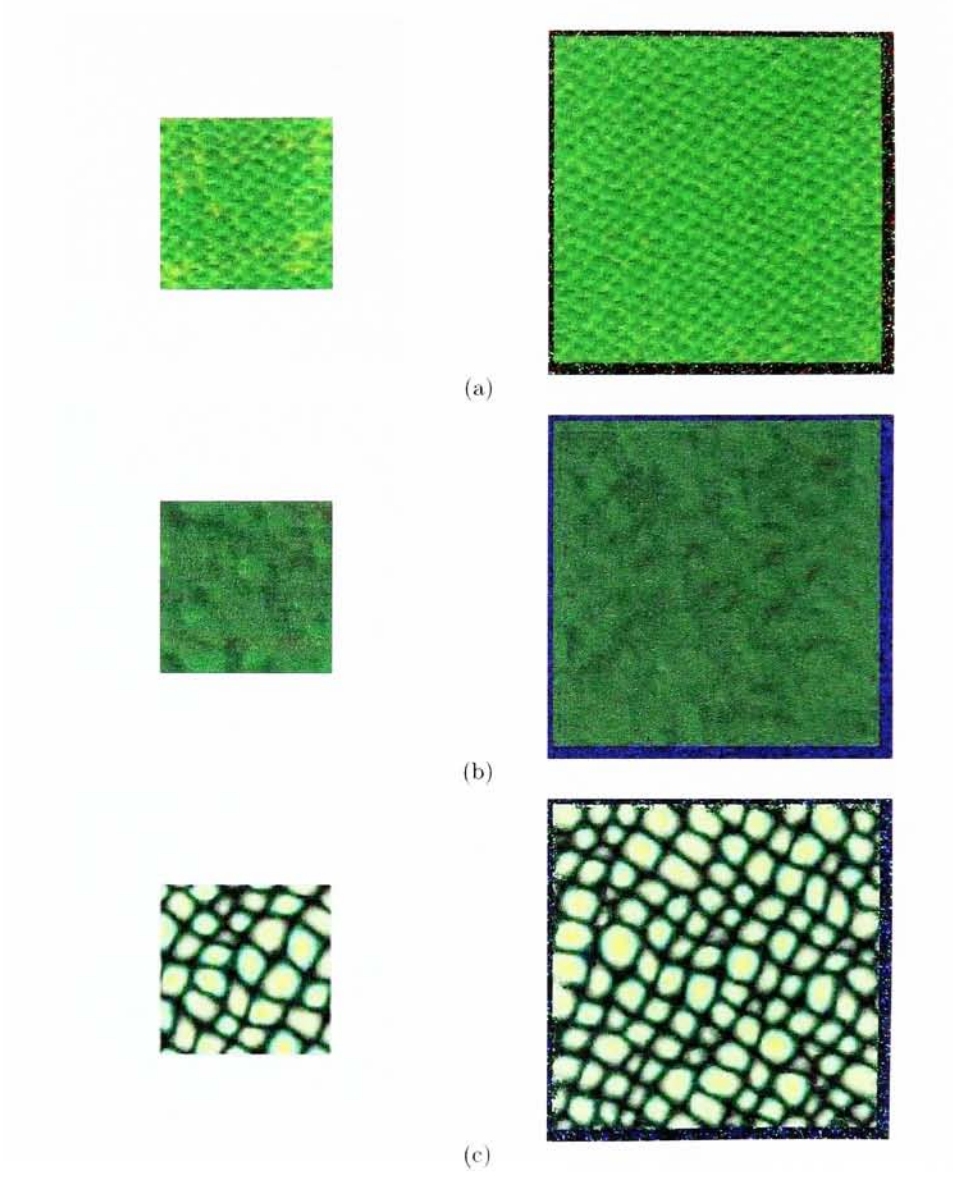


Figure 40: Color synthesis results, resampling using neighborhood size of 16.

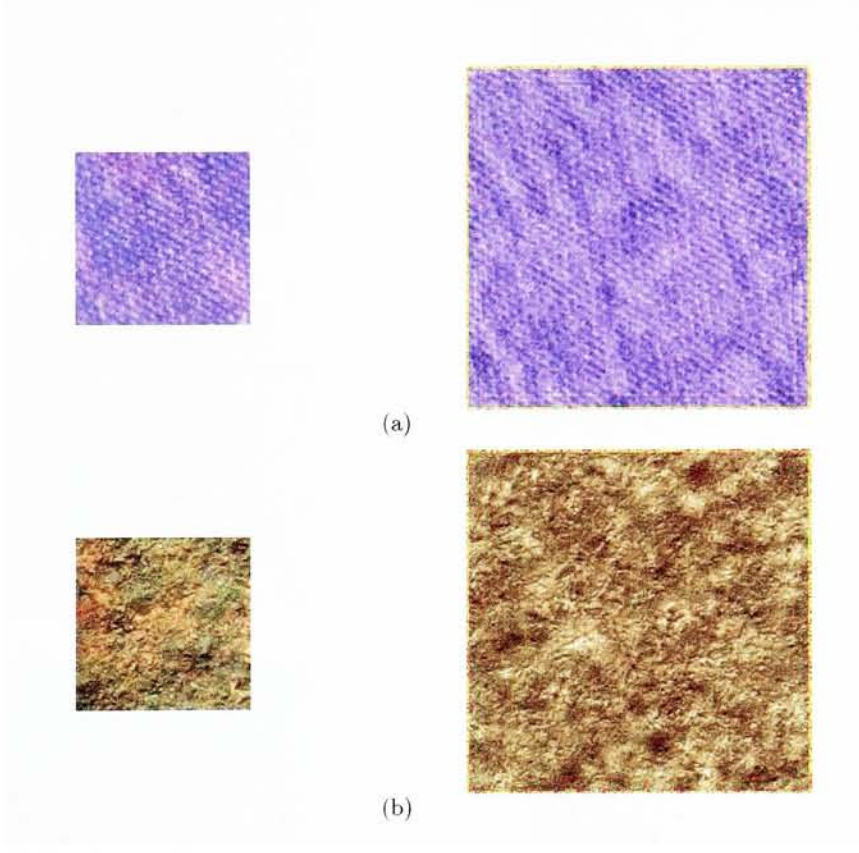


Figure 41: Color synthesis results using P/S method.

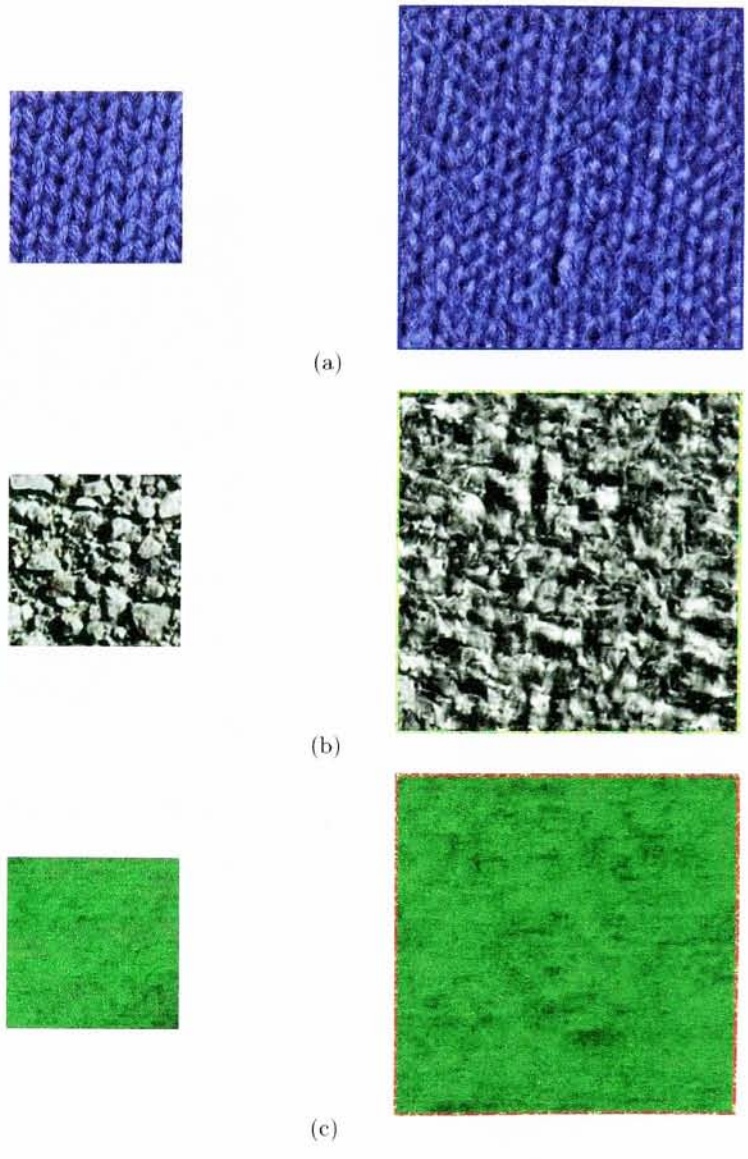


Figure 42: Color synthesis results using P/S method.

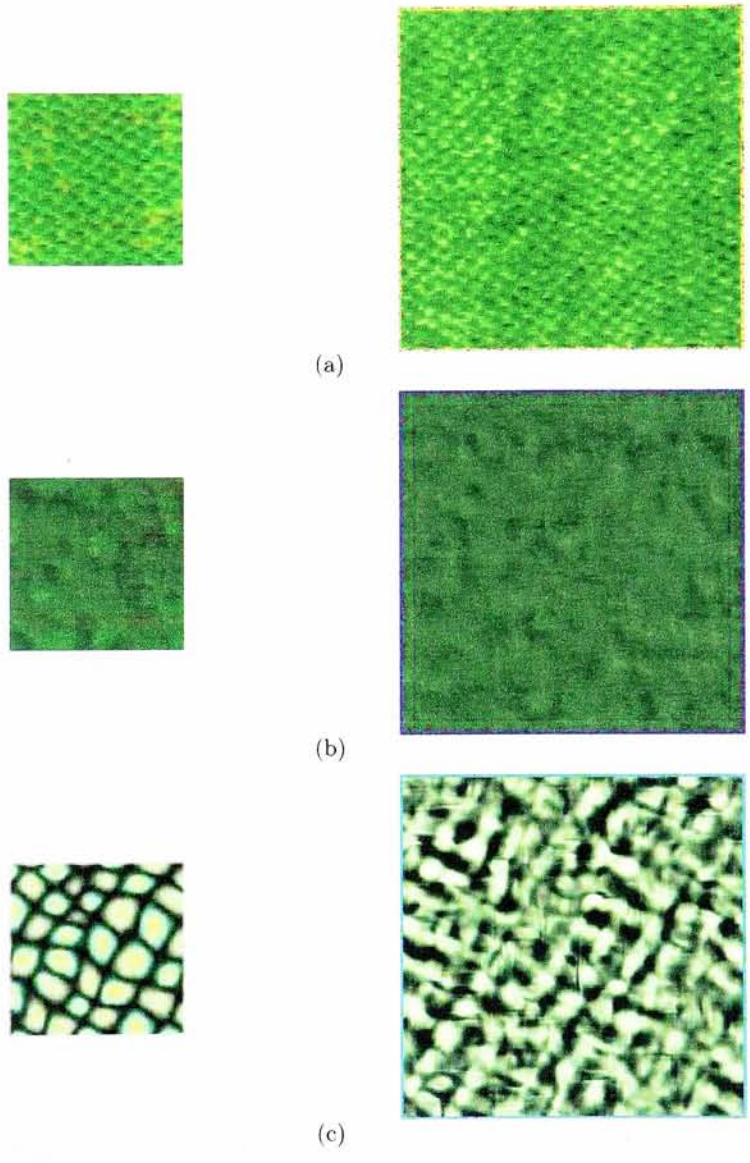


Figure 43: Color synthesis results using P/S method.

7.3 Multispectral Texture Synthesis Results

The multispectral texture used as a test set was collected with an airborne sensor maintained by the Digital Imaging Remote Sensing (DIRS) group at the Rochester Institute of technology using the MISI imaging system. As mentioned, a visual test based on k-means clustering is presented for each algorithm. Although highly qualitative in nature, this sub-space representation is particularly useful since the remaining multispectral results are essentially a set of low order statistics that are known to be necessary but not sufficient conditions for establishing similarity between a sample and synthetic texture. Even though a complete or optimal set of metrics describing texture is an open problem, the argument can be made that the set of all low-order, i.e. first and second order moments, form a natural and coherent set of statistical descriptors suitable for making quantitative comparisons of texture synthesis results. This is based largely on the fact that in many applications where texture synthesis might be used including target detection, segmentation or classification, etc., only the low-order statistics are used. Hence, guaranteeing good agreement between sample and synthetic low-order statistics is of paramount importance.

In addition to the low-order moments, a simple statistical divergence test is used to determine the likelihood that the spectral content of the sample and synthetic textures were generated by the same underlying process. The divergence test chosen is a $Q - Q$ plot of the sample and synthetic spectral distributions. Assuming that the spectra are well approximated as jointly Gaussian, the distribution of Mahalanobis distances [19] over a population should fall into quartiles described by the chi-square distribution. By sorting and plotting this histogram as quartile frequency versus chi-square estimate, a straight line should be observed.

The sub-space projections of the multispectral synthesis results using k-means clustering are presented first. By clustering the spectra of each sample texture, an RGB representation is realized that reflects the spatial distribution of major spectral components. By using the same centroids to cluster a corresponding synthetic texture, a visual comparison can be made in a low dimensional color space. The results of this operation are shown for the four synthesis variants analyzed in the monochrome and color synthesis sections in Figures 44-49.

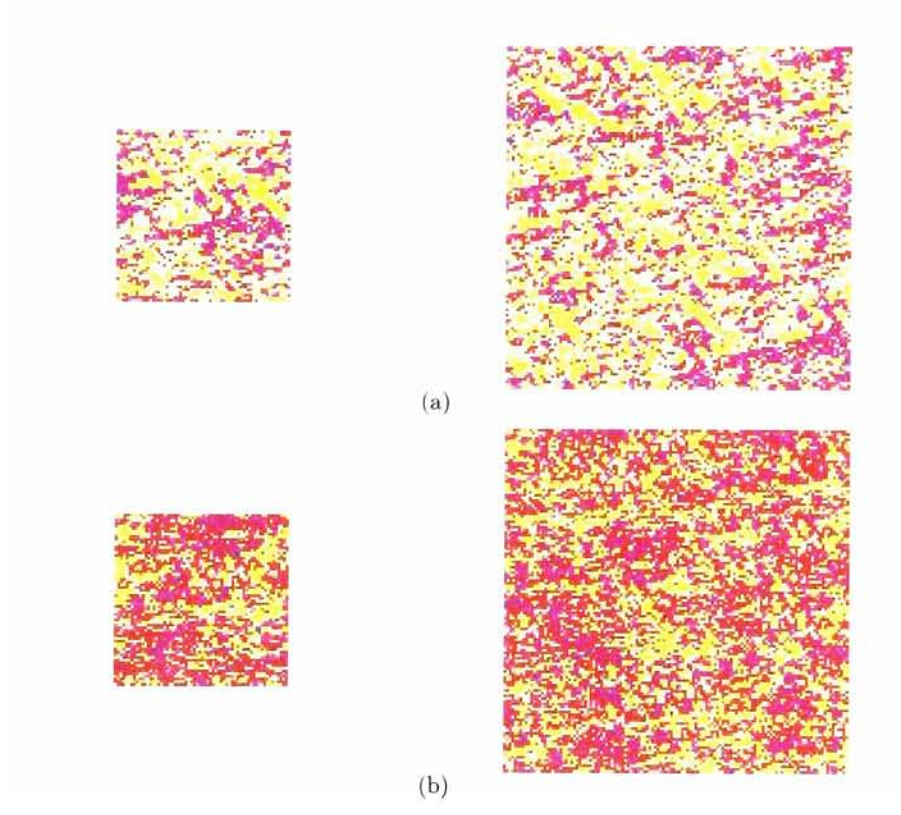


Figure 44: Multispectral synthesis results, quilting with alpha-blend with neighborhood size equal to 16, 22 bands.

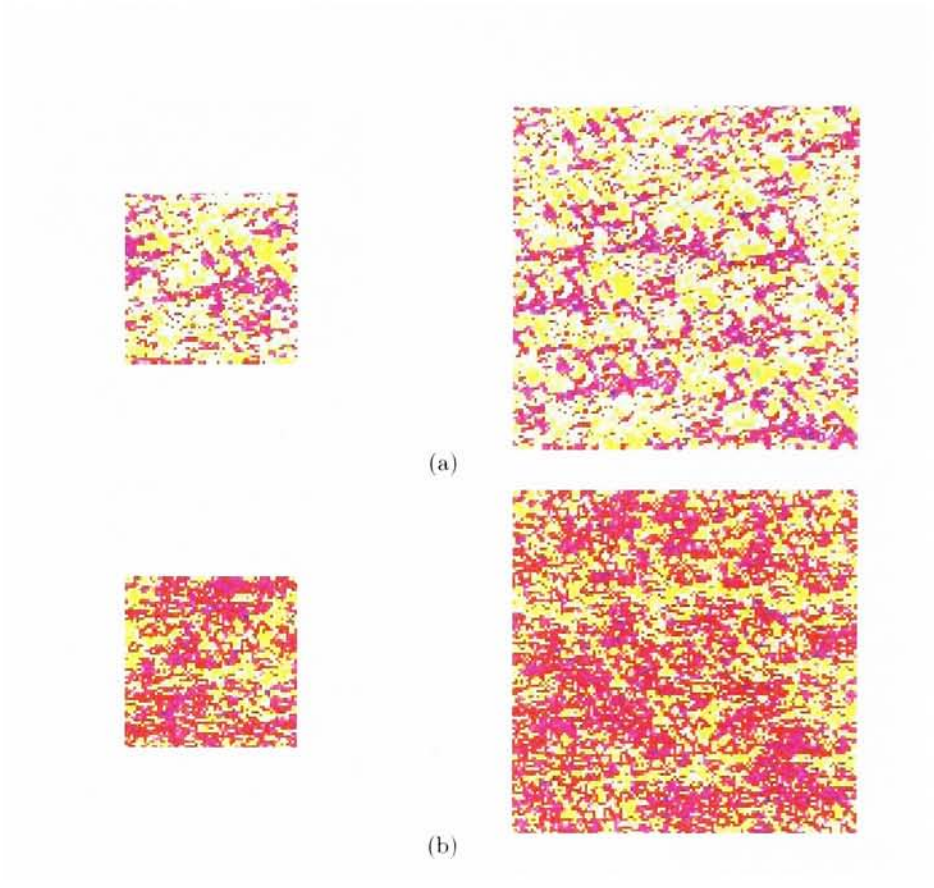


Figure 45: Multispectral synthesis results, quilting with min-cut. a) 22 bands b) 32 bands.

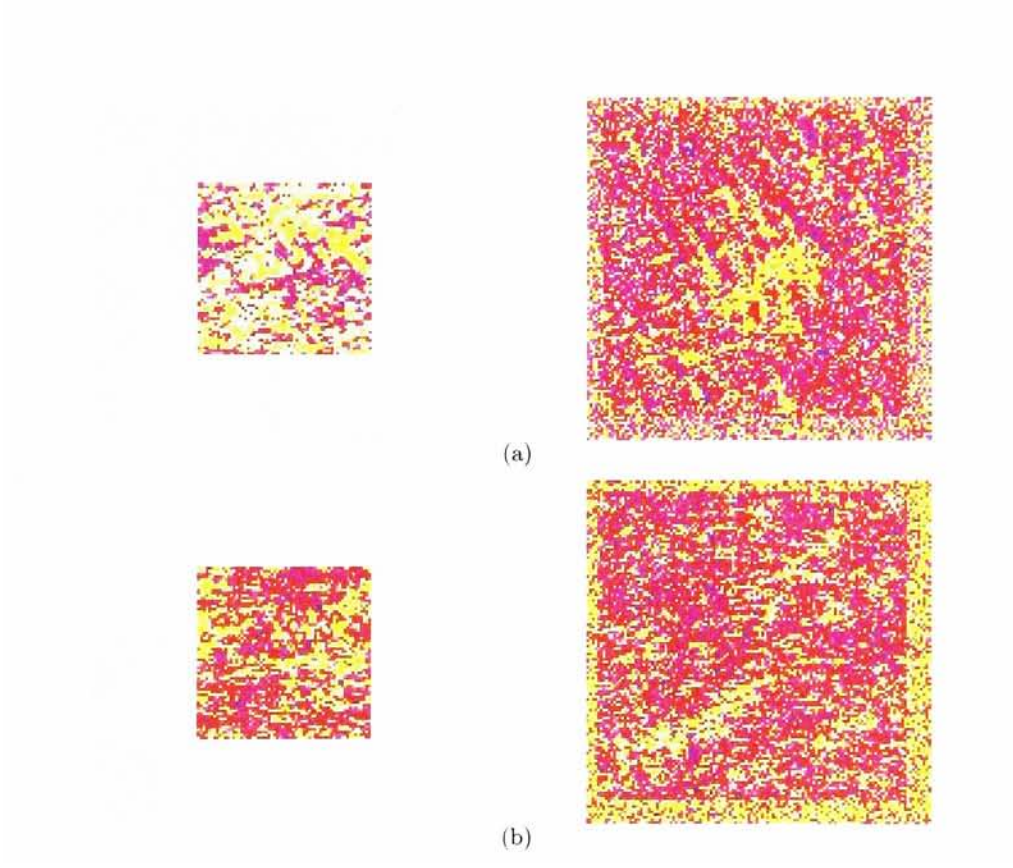


Figure 46: Multispectral synthesis results, resampling. a) 22 bands b) 32 bands.

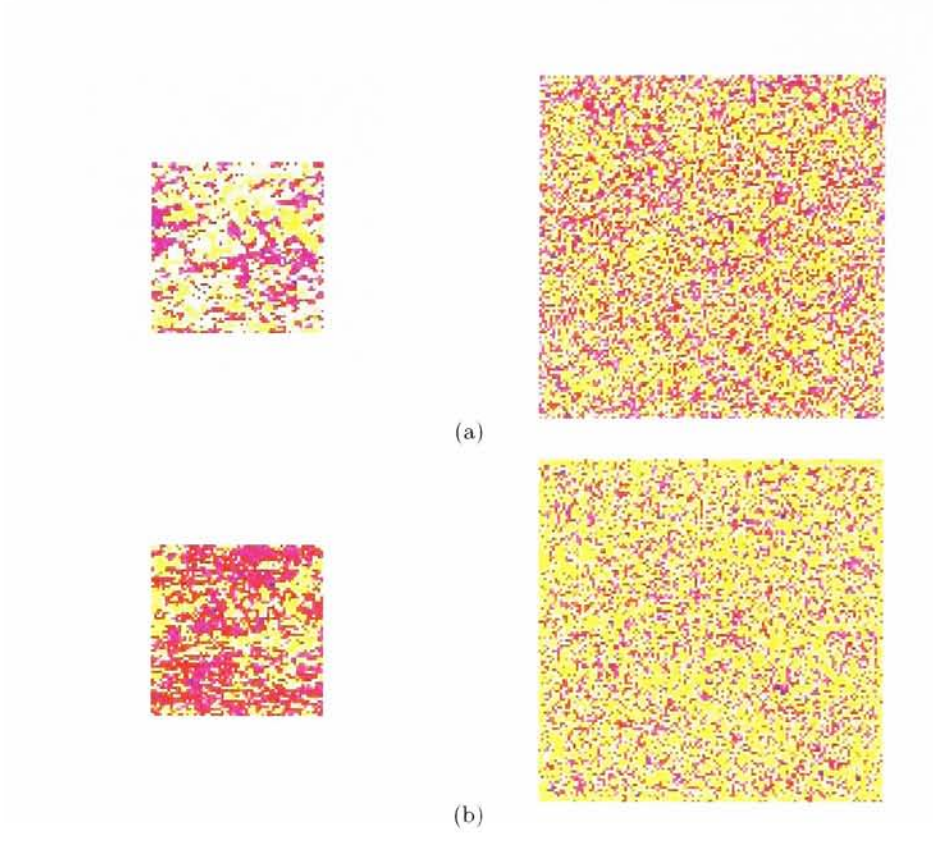


Figure 47: Multispectral synthesis results using P/S method. a) 22 bands b) 32 bands.

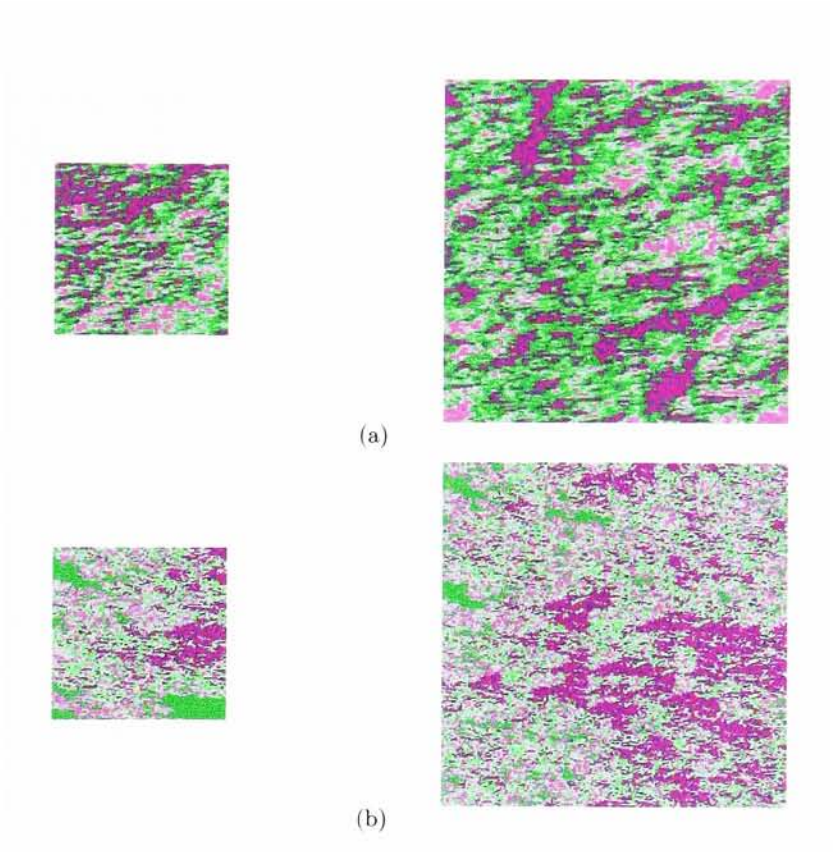


Figure 48: Multispectral synthesis results on 16 bands using a) P/S method b) min-cut resampling.

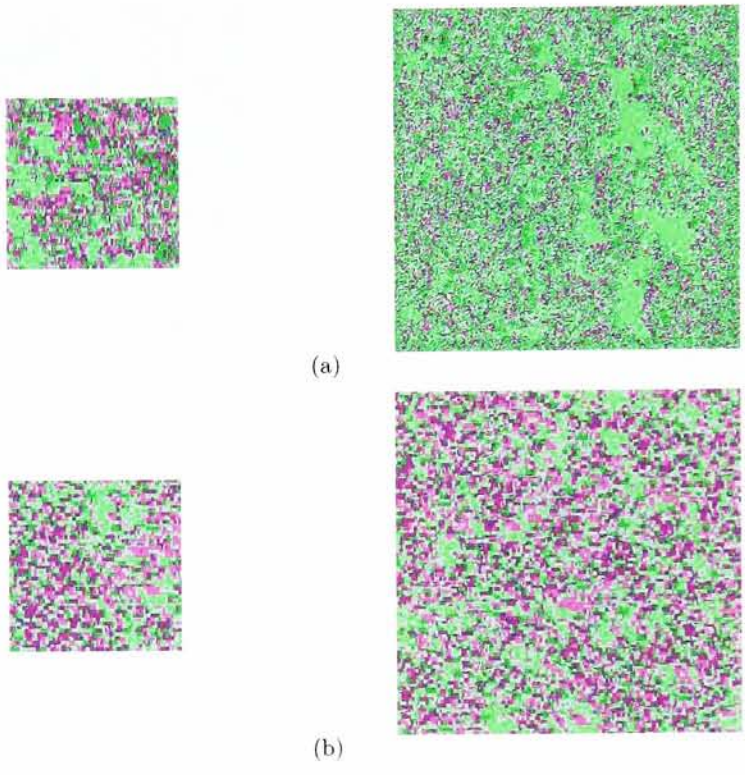
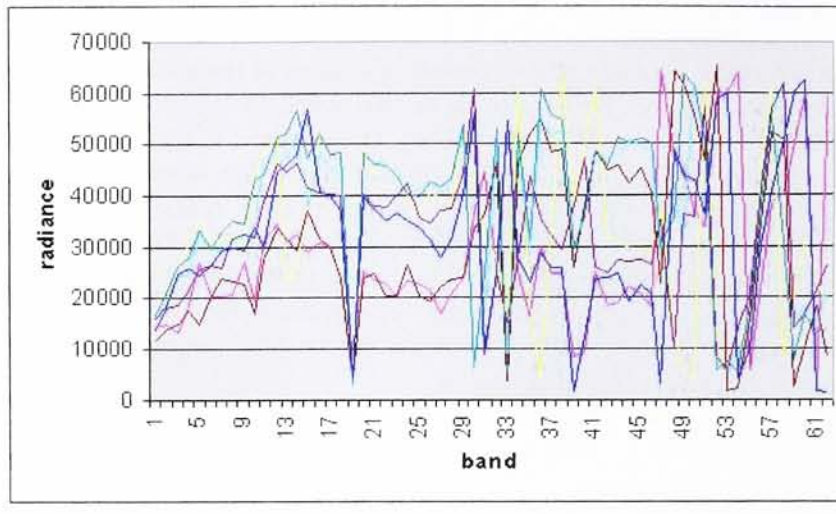


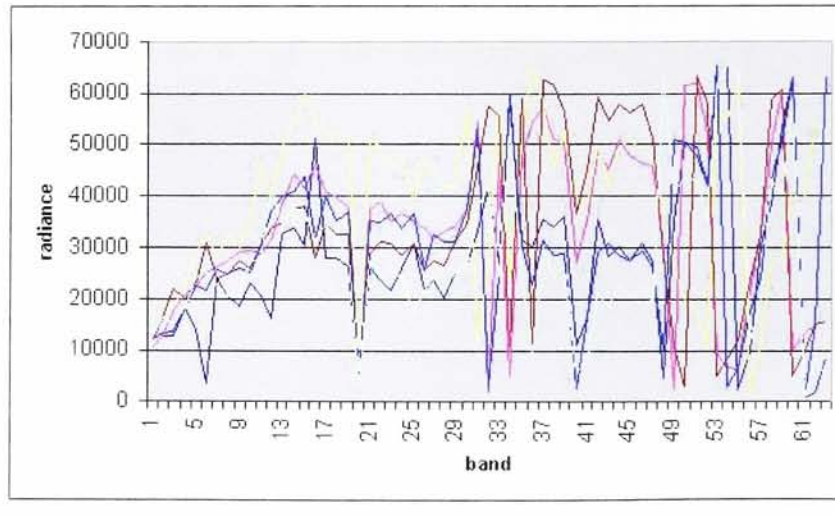
Figure 49: Multispectral synthesis results on 63 bands using a) pixel-by-pixel resampling b) alpha-blend quilting.

Using a simple visual comparison it is evident that each synthesis technique works reasonably well for the multispectral textures presented here. Now that a visual comparison has validated to some extent the synthesis results in the spatial domain, the next step is to observe the spectral content of the synthetic imagery. For this purpose two synthetic textures are selected with their respective spectra plotted vs. the spectra of the corresponding sample texture Figures 62, 63. The spectral curves are selected at random for each image cube. For the purposes of inspection it makes sense to view the spectra of textures generated using alpha-blend resampling and the P/S method but not of textures generated by the other methods. This is because min-cut and pixel-by-pixel resampling copy entire spectral curves directly from the sample texture and the resulting spectra when sampled at random is simply a subset of the input spectra.

By contrast, the P/S method only copies residual spectral information and the alpha-blend quilting method averages the spectral curves that lie in boundary overlap regions. In both of these cases the spectral content of the synthesized image may be considerably different than the sample image. This is especially true when using the P/S method since each spectral curve in the resulting synthetic texture is derived from a coloring transform with a principle component that is uncorrelated with respect to the source spectra.

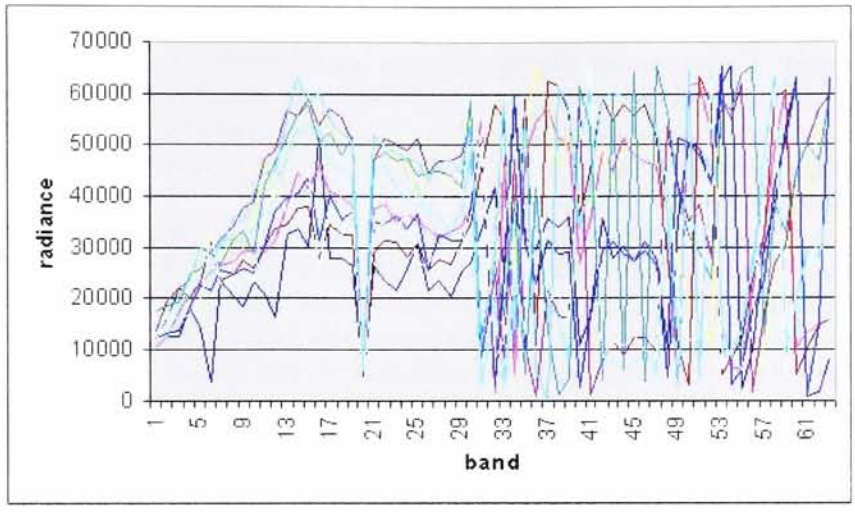


(a)

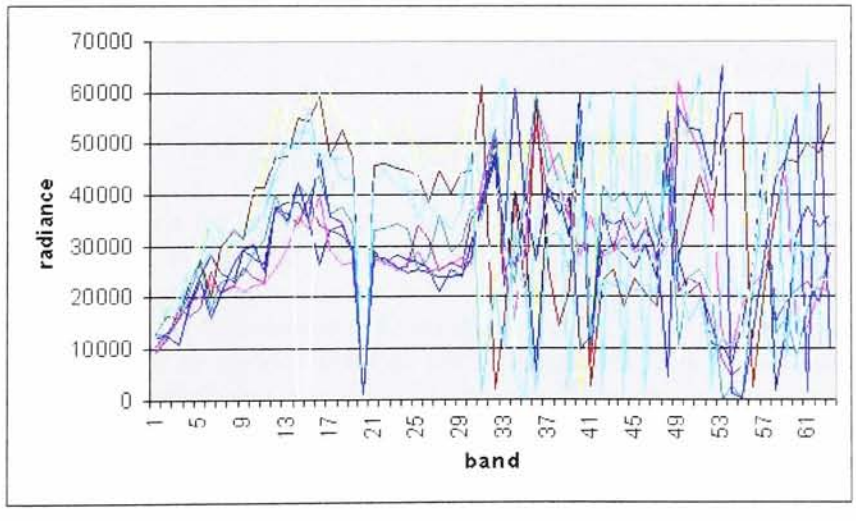


(b)

Figure 50: Spectra sampled from 63 band texture using alpha-blending a) sample b) synthetic.



(a)



(b)

Figure 51: Spectra sampled from 63 band texture using P/S method a) sample b) synthetic.

Referring to Figure 63(b) one notices a single unusual outlying spectral curve that appears to be misregistered with respect to the prominent absorption band. Unfortunately due to the inherent instability of the resampling procedure this type of problem may be difficult to eliminate. Also, the back-projection operation may exaggerate any noise introduced by the resampling procedure. Unfortunately this is a difficult problem to addresss without a new model.

After a visual comparison has been made of the spatial and spectral components of the synthetic textures, the next step is to compare the low-order moments of both a sample and synthetic texture. This comparison is made by calculating the error in the spectral mean and variance of each channel as well as the error in the cross/autocorrelation of each pair of spectral bands. Tables summarizing these results are presented next. The error in the mean and variance is calculated as:

$$1 - |\mu - \widehat{\mu}| / \mu$$

The error in the correlation statistics is reported as a signal-to-noise ratio (SNR) calculated as:

$$20 * \log_{10}(|R_{i,j}|^2 / (|R_{i,j} - \widehat{R_{i,j}}|^2))$$

where $R_{i,j}$ is a correlation function and is calculated by convolving a signal with a mirrored image of itself. Generally the central samples of this function are used to reduce the effects of boundary handling in the discrete convolution. In passing, it is interesting to note that this is the same as the *periodogram* method of estimating the spectral content of a signal. Recall that the power spectrum of a signal is the Fourier transform of its autocorrelation function.

The signal-to-noise ratio is preferred in this case since the correlation error is a function of a *signal* rather than a collection of samples. In order to interpret the SNR results it might be useful to turn to the communications community where SNR values below 20 are generally considered poor and values above 60 begin to become quite good.

Channel	$\widehat{\mu}$	Channel	$\widehat{\mu}$
0	0.994266	11	0.967266
1	0.987678	12	0.97091
2	0.869897	13	0.967472
3	0.931425	14	0.973332
4	0.970631	15	0.964231
5	0.916095	16	0.961645
6	0.850836	17	0.967992
7	0.932231	18	0.972204
8	0.994958	19	0.973327
9	0.981353	20	0.928816
10	0.96554	21	0.951562

Table 3: Error in spectral means, 22 bands using quilting with alpha-blend.

Channel	$\widehat{\mu}$	Channel	$\widehat{\mu}$
0	0.979715	11	0.949248
1	0.98133	12	0.734235
2	0.995027	13	0.976527
3	0.980394	14	0.989835
4	0.998036	15	0.997366
5	0.989327	16	0.995781
6	0.997962	17	0.99661
7	0.985072	18	0.996216
8	0.991782	19	0.999373
9	0.989047	20	0.994142
10	0.997585	21	0.999081

Table 4: Error in spectral means, 32 bands using quilting with min-cut.

Channel	$\widehat{\mu}$	**Channel**	$\widehat{\mu}$
0	0.851425	11	0.699663
1	0.827873	12	0.770052
2	0.801737	13	0.784851
3	0.919219	14	0.792958
4	0.897488	15	0.798475
5	0.923073	16	0.767894
6	0.85309	17	0.800976
7	0.702073	18	0.829927
8	0.657781	19	0.790327
9	0.644021	20	0.765592
10	0.772507	21	0.81381

Table 5: Error in spectral means, 22 bands using resampling.

Channel	$\widehat{\mu}$	**Channel**	$\widehat{\mu}$
0	0.981431	11	0.973749
1	0.951068	12	0.948334
2	0.995574	13	0.941264
3	0.972271	14	0.927957
4	0.97727	15	0.931282
5	0.962206	16	0.93821
6	0.994455	17	0.918889
7	0.93579	18	0.9515
8	0.976991	19	0.963157
9	0.968885	20	0.963366
10	0.924842	21	0.935037

Table 6: Error in spectral means, 22 bands using P/S method.

Channel	$\widehat{\mu}$	**Channel**	$\widehat{\mu}$
0	0.905976	11	0.865774
1	0.955856	12	0.886057
2	0.888975	13	0.938167
3	0.818516	14	0.880585
4	0.844862	15	0.913604
5	0.805383	16	0.901485
6	0.935452	17	0.936737
7	0.728899	18	0.882366
8	0.651222	19	0.977476
9	0.669607	20	0.800945
10	0.918052	21	0.915719

Table 7: Error in spectral variances, 22 bands using quilting with alpha-blend.

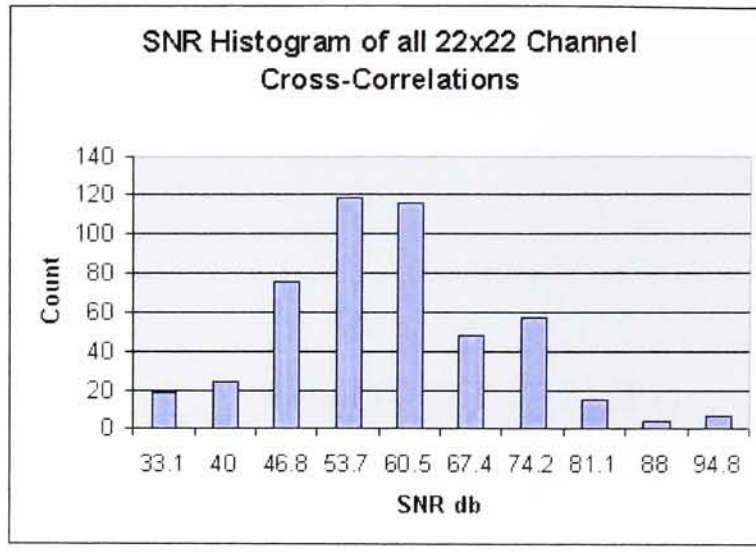


Figure 52: SNR histogram showing distribution of SNR values of all auto/cross correlations between bands using quilting with alpha-blend.

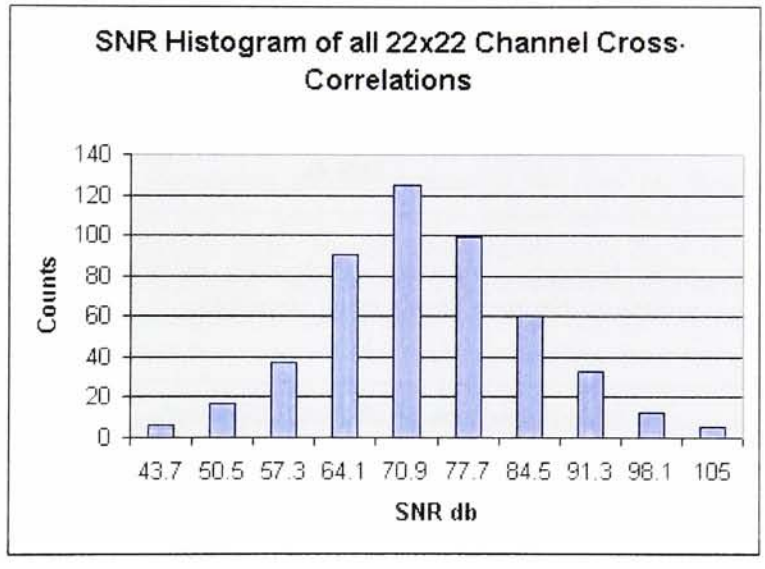


Figure 53: SNR histogram showing distribution of SNR values of all auto/cross correlations between bands using quilting with min-cut.

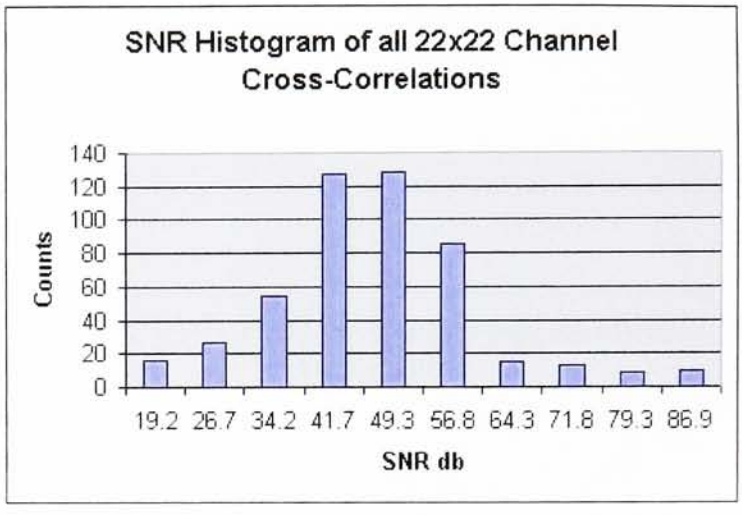


Figure 54: SNR histogram showing distribution of SNR values of all auto/cross correlations between bands using resampling.

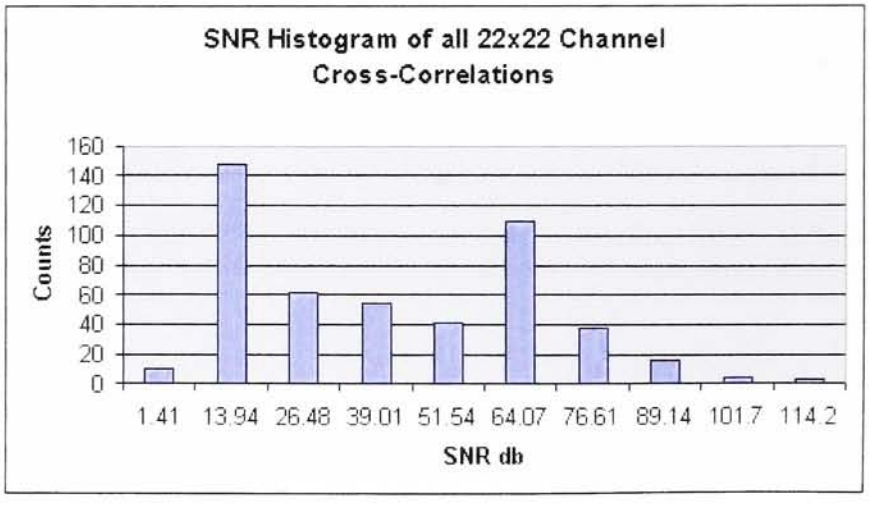


Figure 55: SNR histogram showing distribution of SNR values of all auto/cross correlations between bands using P/S method.

From Tables 3-7 we can see that the synthesized and sample textures agree reasonably well over the set of low order marginal statistics. However, the autocorrelation errors in Figure 55 suggest that the P/S based method is not performing very well. In fact, the results for the P/S method are significantly worse than the quilting based algorithms. The problem may lay partly in the P/S monochrome step or it may be related to the fact that the spectral data is not simply sampled from the source image.

As a final measure of synthesis quality a statistical divergence test is used to determine whether or not the sample and synthetic texture appear to be generated by the same underlying process. The premise of a statistical divergence test is to determine the degree of dependence between two sample populations. Ideally the collection of spectral data from both the sample and synthetic texture should have the same distribution, meaning the individual populations can be pooled to obtain an equivalent population.

Assuming the underlying process generating the spectral data is Gaussian, a $Q-Q$ plot can be used to test the statistical divergence between the sample and synthetic data. As previously mentioned, the sorted quartile distribution of Mahalanobis distances in a sample population generated by a joint Gaussian distribution should follow a chi-square distribution. Using this fact a simple test can be generated that can measure the degree to which a population tends to Gaussian. The test is to plot the quartile distribution of Mahalanobis distances sorted by frequency versus the expected value from a chi-square distribution. Plotting these values should result in a straight line. Now assuming two populations are both Gaussian distributed, then the combined distributions will only be Gaussian if they have the same distribution to begin with. Notice that combination does not imply *addition*. The distinction is important since by definition of scalar and vector addition the sum of Gaussian random variables/vectors is still Gaussian. The results of the $Q-Q$ plots are presented in Figures 56, 57.

In Figures 56, 57 a strong linear trend is apparent. This is a pleasing result but a number of outliers are seen in Figure 56. To access the severity of this result it is important put in the proper context. In this case four of 200 sample points appear to be outliers.

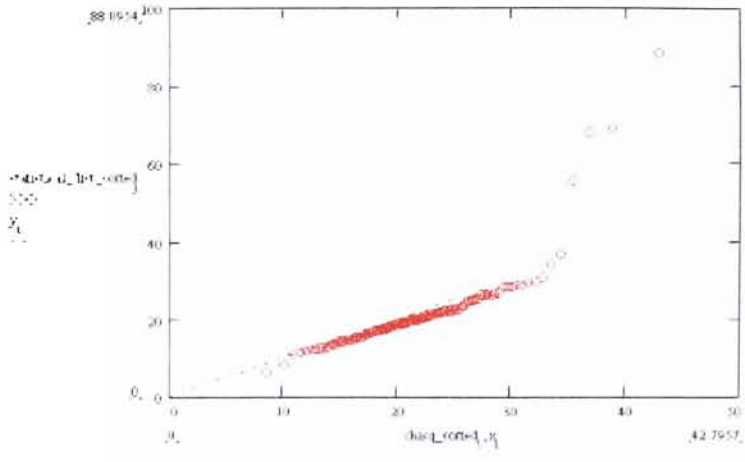


Figure 56: Q-Q Plot of combined sample and synthetic spectra. Using quilting with min-cut on sample texture with 22 bands.

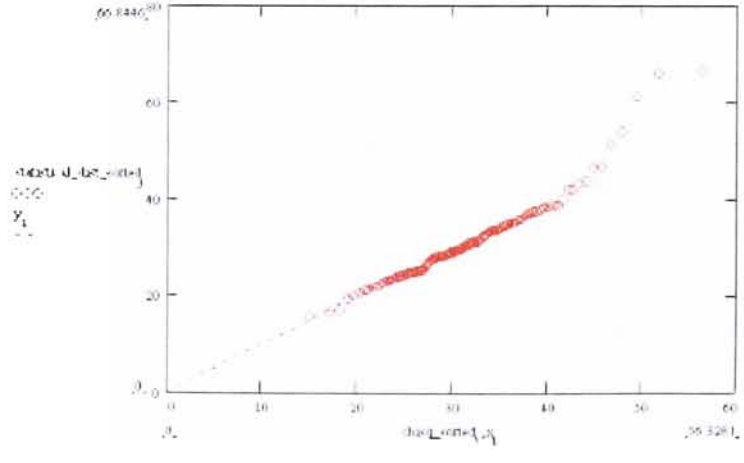


Figure 57: Q-Q Plot of combined sample and synthetic spectra. Using P/S method on sample texture with 22 bands.

7.4 Summary of Results

In order to summarize the results it is useful to analyze them on a number of fundamentally different levels. These include a qualitative visual analysis, statistical analysis where applicable, and an analysis of the theoretical predictions of algorithm performance vs. empirical results. To begin with, a brief discussion of the key aspects of each synthesis algorithm is given. This provides a concise, top-level description of the performance of each approach. This top-level description includes average synthesis timings, the specific class or types of texture that are reproduced well by each synthesis algorithm, memory requirements and stability. Next, each algorithm is analyzed based on their relative merits from a theoretical standpoint including a treatment of the inherent strengths and weaknesses of each approach. Finally, where appropriate, namely in the multispectral case, the statistical results are analyzed.

7.4.1 Performance Matrices

In order to convey a good top-level description of each algorithm, a performance matrix is presented summarizing performance characteristics for the separate tasks of monochrome, color, and multispectral texture synthesis. Each matrix includes asymptotic time complexity, average timings, memory requirements, stability, and a breakdown of performance of three classes of texture: stochastic, structured and mixed. A scale ranging from $++$ to $--$, is used to rank the optimality of each algorithm based on the aforementioned criteria. An additional asterik (*) indicates that the results may be dependant on operator tuning. Also, in interpreting the time complexity figures N refers to the size of the synthetic image and S is the sample size, i.e. $128x128$ etc., and n is the size of the quilting region. These matrices will form the basis for a more complete treatment of algorithm performance especially in cases where an extremum is present.

It summary, the performance matrices indicate that quilting has very good overall performance characteristics. Resampling does reasonably well for a variety of texture classes but exhibits serious stability issues. One thing that is quite noticeable is that the Portilla/Simoncelli method appears to perform somewhat poorly. However, using such a coarse analysis based on empirical results undermines the important property of low-order statistical constraint enforcement associated with the Portilla/Simoncelli method.

Algorithm	Quilting AB	Quilting MC	Resampling	P/S
Time Complexity	(N/n)(S)log(S)	(N/n)(S)log(S)	NSlog(S)	(N)log($N^{1/2}$)
Average time	126 sec	141 sec	289 sec	388 sec
Memory Usage	3.5 MB	3.5MB	4.1 MB	2.2 MB
Stability	++	++	−−	+/−
Stochastic	+/−	+/−	++	+/−
Structural	+++*	+++*	+/−	+/−
Mixed	++	++	+/−	+/−

Table 8: Performance matrix summarizing monochrome texture synthesis performance characteristics.

Algorithm	Quilting AB	Quilting MC	Resampling	P/S
Time Complexity	(N/n)(S)log(S)	(N/n)(S)log(S)	NSlog(S)	(N)log($N^{1/2}$)
Average time	126 sec	141 sec	289 sec	388 sec
Memory Usage	3.5 MB	3.5MB	4.1 MB	2.2 MB
Stability	++	++	−−	+/−
Stochastic	+/−	+/−	++	+/−
Structural	+++*	+++*	+/−	+/−
Mixed	++	++	+/−	+/−

Table 9: Performance matrix summarizing color texture synthesis performance characteristics.

Algorithm	Quilting AB	Quilting MC	Resampling	P/S
Time Complexity	(N/n)(S)log(S)	(N/n)(S)log(S)	NSlog(S)	(N)log($N^{1/2}$)
Average time	126 sec	141 sec	289 sec	388 sec
Memory Usage	3.5 MB	3.5MB	4.1 MB	2.2 MB
Stability	++	++	−−	+/−
Stochastic	+/−	+/−	++	+/−
Structural	+++*	+++*	+/−	+/−
Mixed	++	++	+/−	+/−

Table 10: Performance matrix summarizing multispectral texture synthesis performance characteristics.

7.4.2 Critical Analysis of Quilting Method

Since quilting with alpha-blend, min-cut and resampling are basically variants on the same class of model they share a number of important properties. To begin a discussion of these properties the focus will be on monochrome texture synthesis. Since a single monochrome synthesis step is used in the generalization of quilting to higher spectral dimensions this is a natural starting point for a critical discussion of the quilting method. This analysis will ultimately lead to an important theoretical result concerning the *convergence in distribution* between sample and synthetic spectral information. Finally, it will be demonstrated that this theoretical result is well supported by the empirical evidence.

It has already been stated that quilting should display much better stability than pixel-by-pixel resampling. Clearly the experimental results validate this intuition. To restate this intuition, consider how quilting differs from resampling. In the resampling case, a number of complexities must be resolved that are not present in the quilting variant. These include boundary handling, effective seeding, and neighborhood determination including both size of shape. Quilting avoids any boundary handling issues, a seed is determined quite easily as an entire block of sample texture and neighborhoods are determined by boundary overlap. Ultimately this reduces the number of tuning parameters needed to establish good synthesis results to just the determination of an appropriately sized neighborhood.

Since quilting transfers a significant amount of information to the synthetic image at each step, any noise in the boundary overlap associated with a suboptimal synthesis step is 1) distributed over the entire boundary 2) small relative to the size of the quilted block 3) directly affects at most two subsequent synthesis steps. The last property is likely the most important in the context of stability. Contrast this to the resampling case and we see that a single suboptimal synthesis step has a significant impact of numerous subsequent synthesis steps. Of course quilting suffers from other problems. The most significant is that given a finite texture sample, the probability of finding similar regions of overlap in a sample image gets small as the quilting neighborhood grows in size. This means that it might not be possible to find a rich set of candidate synthesis blocks in a sample image. This is exactly the kind of problem we see in the synthesis results. Resampling naturally avoids this since the probability of finding similar individual pixels for a given neighborhood does not decrease as quickly as neighborhood size increases.

This last result follows directly from the axioms of probability that state that the probability of a joint event is bounded by the minimum probability of either event:

$$P(A, B) \leq \min(P(A), P(B))$$

In the context of resampling, this result suggests that it is more probable to find *a single highly likely pixel* given a set of neighbors than it is to find *a set of highly likely pixels* given their common neighbors. In summary, there is trade-off between stability and maintaining a rich and diverse search space.

When texture is not monochrome, stability has important effects on the spectral characteristics of the resulting synthetic texture. Quilting is a sampling procedure drawing from an example texture. If this process becomes unstable three things can happen. First, the resulting sampling becomes random failing to capture any spatial correlation. Second, in the other extreme, the spatial correlation becomes deterministic when the search process gets stuck in a particular region of the search space. A third possibility is that the sampling process doesn't degenerate to a deterministic process but the spatial correlation introduced is not representative of the sample texture. Given these three possibilities it is quite easy to describe with certainty what will happen to the resulting spectra in the limit of large example textures.

The argument is quite simple. Given sufficient sample data, i.e. for large sample textures, a quilting neighborhood can be chosen so that in the limit the probability that population distributions are different between the sample data and the quilted data zero. Stated in a different way if we can take large enough samples from a large population, we know that any statistics estimated from the sample will converge to the underlying population. The proof is by (weak) Law of Large Numbers (LLN) and is readily expressed in terms of the Cauchy convergence criteria of two random sequences $|X_n - X_m| - > 0$ as n and m get large. Of course this result says nothing about the spatial correlation in the texture but it does establish an important theoretical result nonetheless. In fact this result is enough to prove *convergence in distribution* over the spectra in a sample and synthetic texture for large samples. By definition convergence in distribution means that a random sequence approaches a single limiting random variable with a given distribution. The central limit theorem is an example of convergence in distribution. The question now is, how large does the sample need to be

until good convergence is realized? Initial results suggest that this size is well within reasonable limits for a practical application. Referring to Figures 56, 57 one observes that the distribution of Mahalanobis distances around a common mean exhibits strong linear correlation with the predicted chi-square distribution. Recall that this result suggests that the joint distribution of radiance values in both the sample and synthetic texture have in fact converged to the same distribution.

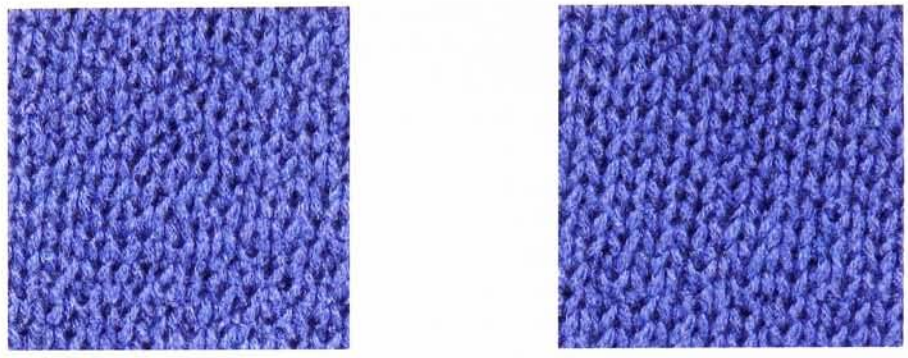
Within the context of this theoretical limit it is possible to completely describe the ways in which a quilting procedure will fail to sufficiently reproduce the correct spectral distribution. Consider the first case where the sampling procedure fails to introduce any spatial correlation, i.e. the sampling is random. In this case the spectral statistics are guaranteed to converge via LLN as we saw before. Of course there won't be any spatial correlation and this would arguably not suffice as a sufficient texture synthesis procedure.

Now consider the case where the sampling operation fails to capture the inherent spatial correlation in a sample texture but does not degenerate to a deterministic process. In other words, the sampling process is somewhere between completely random and completely deterministic. Here again LLN will guarantee convergence of the spectral distributions.

Finally, there is the case when the resampling procedure becomes deterministic. If texture is generated pixel-by-pixel there is virtually no possibility of convergence given the sample spectra are random enough. However, if sufficiently large texture blocks are quilting together the fact that the blocks are selected in a deterministic manner does not adversely effect convergence. This is fairly obvious since a single block of texture will carry all the information contained in the original sample provided the sample and block are sufficiently large. This is the definition of ergodicity, which has already imposed as a requirement on the underlying random process modelled by the various synthesis techniques presented in this work.

In contrast to quilting using min-cut and resampling, the alpha-blend variant introduces noise in the spectra. The noise is associated with local averaging of spectra at the boundary overlap. The addition of this noise affects the convergence of sample and synthetic spectral distributions. In the limit this noise becomes negligible as neighborhood sizes increase, but this fact may effect how quickly good convergence between the spectra in a sample and synthetic texture can be observed in practice.

One unfortunate aspect of the quilting variants is that they rely heavily on operator input to obtain good synthesis results in some cases. This is



(a) (b)

Figure 58: Texture synthesis results after tuning neighborhood size using quilting with min-cut. Previous result is on left using neighborhood size of 16. On the right the neighborhood is increased to 32.

demonstrated on a sample image that was not reproduced well using default settings. Clearly the new synthesis results are much better after tuning the algorithm to match the characteristics of a particular texture sample Figure 58.

In certain applications it might not be possible to tune a synthesis algorithm to individual classifiers. Imagine the daunting task of tuning a synthesis algorithm to effectively synthesize textures from a large texture library. This type of problem arises in synthetic scene generation when a large number of disparate texture patches are required to render a scene. What's needed in this case is an automatic method that has good performance characteristics. This is where the P/S method provides a possible solution.

7.4.3 Critical Analysis of Portilla and Simoncelli method

As was previously mentioned, the Portilla/Simoncelli method attempts to find a complete set of basis functions and statistical descriptors in order to represent arbitrary textures by a single unified model. The results suggest that the P/S method falls short of this idealized model. This suggests that a low order approximation to the underlying probability distribution that describes an ensemble of similar texture images may not suffice in some cases. But this should be not be too surprising since the statistical constraint enforcement operations performed in the P/S method are not

inherently different from other low-order approximations based on Fourier coefficients, Karhonen-Loeve coefficients, or Taylor series expansion. In applications where the agreement over this set of low-order statistics is all that is required, the P/S method is an obvious choice. In fact, the enforcement of first and second order statistics are often sufficient for many models used in vision and image processing applications.

A particular strength of the P/S method is that it doesn't require any operator tuning, which makes this method appropriate for certain applications where quilting simply cannot be applied. For instance, when large databases of texture must be synthesized precluding the selection of a quilting block-size for each texture. In addition to this strength, the P/S method is particularly useful for synthesizing textures that can be modelled as a particular subclass of random processes known as wide-sense stationary processes.

A wide-sense stationary (WSS) process has a constant mean and shift-invariant correlation function. The enforcement of a constant mean value and the autocorrelation function between a target and synthetic texture can be made exact since since $R_{XX} = K_{XX} + \mu_X^2$, where R is the correlation function and K is the covariance function which is invariant to the mean. This is in contrast to the P/S method that uses a multi-scale iterative procedure to enforce the mean and autocorrelation that may not converge. However, the trivial multiresolution representation of a texture is simply the texture itself, and the P/S method can be modified to perform an exact constraint enforcement of the mean and autocorrelation functions by using a pyramid with one level. Thus the P/S method has the attractive theoretical property of being able to enforce wide-sense stationarity. This is important in a number of practical applications were random signals are modelled as WSS.

However this is a somewhat limited, although potentially useful, result for applications that only require monochrome texture synthesis. Unfortunately, it is a difficult task to extend this result to multiple channels. The current method using resampling to fill in the remaining spectral channels can't guarantee that the resulting texture will display the correct spatial and spectral correlation (where correlation refers specifically to the correlation function).

It is important to remember that the Portilla and Simoncelli method, as extended to color and multispectral texture synthesis, is strongly related to quilting. To begin with, the same generalized Markov Random field model is inherent in each approach. The only difference is that the spatial correlation is captured by a different model. To illustrate this point further,

consider exchanging the single monochrome texture synthesis step using the P/S method with a quilting or resampling procedure. The subsequent resampling operation used to fill in the remaining spectral channels is independent of the method used to generate the spatial information in the single principle band. Since resampling is used to generate the spectral information in all the techniques discussed in this work, it is only natural to assume that the performance of each approach will be tightly coupled.

It is also important to emphasize that although the resampling step used in quilting and P/S multi-channel texture synthesis are related, they are not equivalent. The major difference is that quilting copies entire spectral curves from a sample texture whereas the P/S approach simply copies residual spectra. It has already been noted in the results section that this might have undesirable effects related to the back-transform operation used to recorrelate the spectral bands. In short the back-transformation is essentially a one-to-one mapping when using quilting but can produce erroneous curves in the P/S framework.

It is interesting to note that the P/S method works extremely well in certain cases and fails totally in others that appear to have similar characteristics, e.g. two structural textures with similar complexity. This is in contrast to the quilting methods that seem to have predictable performance in the sense that one can get a good feeling for which textures will work and which will not.

One reason for this is that the input textures might not be of sufficient size. The estimates required for statistical constraint enforcement may be sensitive to the relative sample size. This problem is likely worsened by the downsampling operations performed by the multliresolution decomposition. At the upper level of the sample pyramid, parameter estimates may exhibit large variance due to the lack of sufficient sample data.

As a general summary of the P/S approach it seems to be inferior in a number of ways to the quilting method. First, the resampling procedure used to fill in the residual texture channels doesn't display the pleasing result of convergence in distribution based on the central limit theorem i.e. the Law of Large Numbers, that was previously discussed. Second, the algorithm has its own stability issues that are likely only compounded by the subsequent resampling operation.

8 Conclusions

In summary, this work demonstrates a plausible framework for extending state-of-the art monochrome texture synthesis algorithms to handle texture with arbitrary spectral dimension. A particular distinction of this work is that this has been accomplished in the context of limited prior work. What's further is the fact that much of the previous work has focused specifically on separating spatial and spectral modelling using various multi-channel color space models. But this approach has been shown to have dramatic weaknesses that may interfere with a generalization to textures with high spectral dimension. As a result, this work represents a new contribution in the area of texture synthesis that may be valuable for a number of practical applications where high quality multispectral synthesis results are needed.

Specifically, the notion of an abstract view of the synthesis process is shown to be practical and effective. Under this abstract view, the specific spectral information at each pixel location is independent of the spatial model used to capture correlation over the ensemble of spectral images representing a specific texture. By using a simple color space transform the spatial information is condensed to a single channel that can be used to condition the placement of the remaining residual spectral information. Using a low dimensional representation of the spectral texture allows for the use of existing synthesis techniques.

The synthesis results presented in this work suggest that this approach is reasonably well formed. In fact, a simple analysis of the nature of a 2D vector random sequence that is both stationary and ergodic provides additional support that the approach taken in this work is valid. First, consider the statistical specification of a vector random sequence:

$$f(\overrightarrow{x_i}, \overrightarrow{x_{i+1}}, ..., \overrightarrow{x_{i+N}}; 0, 1, ..., N)$$

where each $\overrightarrow{x}$ is a vector quantity at a position relative to time 0. As N goes to infinity the function $f(...)$ specifies a probability distribution for the entire random sequence. If the random sequence is stationary and ergodic, the distribution can, by definition, be estimated from a single sample provided it is sufficiently large. Extending this statistical specification to 2D, i.e. a multispectral texture, is straightforward and is fundamentally no different than the specification of the 1D sequence. Now, given the statistical specification of a random sequence it is theoretically possible to sample it

in order to derive novel instances of a multispectral texture. Unfortunately this is the where a number of practical problems begin to be encountered. Most importantly is the fact that sampling and arbitrarily complex probability distribution is not straightforward. In fact, in the general case, it can only be done through random search, e.g. simulated annealing, Gibbs sampling, or Monte-Carlo methods. These methods are not only burdensome computationally, sometimes prohibitively, but they also fail to provide guidance as to how well the resulting samples actually represent the underlying random process. Contrast this with sampling a Gaussian distribution where the mean value is basically the prototypical sample and would be expected to be highly representative of the underlying sample space from which the distribution is derived.

To summarize thus far, under the ideal conditions that a stationary ergodic 2D random sequence is given for which a statistical specification is available, there is often no practical way to sample the distribution. This is obviously not a good situation and the reason why texture work based on a statistical specification of the process has all but been abandoned. Instead, current research focuses on two distinct approaches. The first is based on low order moments that approximate a distribution function. The second is the graphics based approach known as resampling. Both of these approaches are sound and can be related directly to the problem of sampling an arbitrarily complex distribution function describing multispectral texture.

In the first case, namely low order statistical constraint enforcement, the principles of probability theory can be used directly. The Fourier transform of a probability distribution is called a characteristic equation. The characteristic equation is also known as a moment generating function since its derivatives have a one-to-one correspondence to the distribution's moments. This means it is possible to describe a probability distribution by its infinite set of moments. As is often the case when functions are described by an alternate set of basis functions, an accurate low-order approximation may be possible using only a subset of these basis functions. In this case, the set of low order statistics including mean and autocorrelation might suffice. Also, while a random process might not in general be ergodic, it may be ergodic in its low-order moments. This means that accurate estimates of the low-order moments may be possible given finite sample data. In short, using low order approximations of a probability distribution is a theoretically sound approach. Extending this notion into a complex synthesis framework like that used by Portilla and Simoncelli is simply a straightforward extension of

these principles.

In the resampling framework, rather than searching over the space all possible sample functions until one with a sufficiently likely configuration is found, the search is performed over an input image that is already a likely configuration. Obviously these two operations are somewhat closely related. The main difference is that rather than providing an explicit posterior probability measure, resampling relies on a geometric similarity measure. In other words, rather than computing the posterior probability of observing a texture sample drawn from a given distribution, resampling uses a geometric criterion for determining the likelihood of observing the resultant texture. The difference between these two approaches is easily illustrated in the context of a nearest neighbor distribution. For this purpose of illustration this discussion will deal with monochrome texture but the extension to higher spectral dimension does not result in any loss of generality.

First consider how Gibbs sampling or simulated annealing produces a texture. Basically for each pixel location in a synthetic image there is an associated likelihood of observing a pixel value based on the surrounding pixels. Using the concept of a Markov random field, namely, the pairwise correlation of pixel values goes to zero over large spatial extent, the observed likelihoods over an entire image can be combined into a posterior probability measure. For those who are interested refer to [6] and [12] for a discussion of MRF models, cliques, and Gibbs-Markov equivalence to further understand this process. The search used in Gibbs sampling simply changes pixel values at random based on whether or not the resulting posterior probability is enhanced by the change. In this sense Gibbs sampling works based on the principle that if the localized constituent pieces of the texture are highly likely based on a given distribution, then the entire texture will also be likely.

Now consider the resampling model. Basically, rather than randomly changing pixel values, information from a sample image is used to build a new texture that should also consist of highly likely localized constituent pieces. In the resampling case the likelihood of the constituent pieces is no longer guided by an explicit distribution. Rather, it is defined implicitly by a non-parametric estimate of an equivalent MRF based on nearest neighbors.

At this point it is easy to show that the two approaches produce equivalent results as the number of sample points from which to choose nearest neighbors gets large. To see why consider the following example. To begin with, a finite of set of points in a distribution space is provided where the actual probability mass of observing those points is known. Now assume that this probability

mass can closely approximate the density in a small region surrounding a given point. So if a point is tested within this small region an accurate probability estimate could be furnished. The only real problem begins to emerge when the test points are outliers.

This problem can be overcome as the number of known points increases to infinity. In this case the distribution becomes continuous and there is no longer a concept of an outlier. This is because in the limit there will always be a large number of neighbors around any test point that can furnish reliable density estimates. Incidentally, this concept is directly related to the method of Parzen windows [19]. Of course such a limit will never be reached in practice but empirical results suggest that good results are still achieved with limited sample data.

Each method presented in this work has distinct weaknesses and advantages. Taken in whole, the suite of synthesis tools that have been developed are shown to successfully synthesize an extensive variety of texture classes with arbitrary spectral dimension. Although some synthesis failures are evident, the primarily graphics based approaches used in this work are shown to reproduce spectral statistics almost flawlessly. The main motivation for using a graphics based approach is to ensure that, generally speaking, the synthesis results won't deviate far from the input in a statistical sense. Using this fact an important theoretical result is developed. Specifically, using the Law of Large Numbers, it can be proven that the statistical characteristics of the spectra in a sample and synthetic texture can always be shown to converge given sufficient sample data.

The timeliness and memory consumption of each algorithm are well within bounds appropriate for a standard desktop workstation. This is another important result especially when compared to prior work in the literature. Direct application of Markov Field models in previous work has required massively parallel architectures. Nave implementations of resampling could potentially take many hours to complete. Other attempts to optimize the nearest neighbor searching process have generally used sub-optimal greedy search techniques based on sub-space projections. The kd-tree implementation used in this work is in contrast optimal, and still very fast with relatively small memory footprint.

Generally speaking, the quilting methods appear to work better than the P/S model. This is largely due to the inherent deficiencies in the wavelet representation used by the P/S model to adequately describe a sample texture. The strictly graphics based approach of quilting is shown to have very

good reproducing characteristics. It is not unreasonable to argue that the low bias of such a non-parametric model is highly effective at representing the statistical characteristics of an input texture especially when confronted with sparse data. Of course the P/S method has an important theoretical advantage since it can basically guarantee wide sense stationary agreement with the underlying process generating the sample data.

9 Extensions

Using the techniques outlined in this work a number of interesting extensions are possible. The first is simply a different type of synthesis operation called hole-filling. In this configuration, the task is to replace a region of texture that is otherwise missing from a sample texture. This is useful for operations that involve masking out objects in a scene and replacing them with a continuation of the background texture.

The second extension is a fundamentally new concept that focuses on extending the spectral content of a multispectral image. Using statistical techniques, the idea is to generalize the notion of multispectral texture synthesis to include the spectral domain. This is a natural extension to the idea of multispectral synthesis since the spectral content of a texture itself might be the primary interest. Using this extension it becomes possible to synthesize both spatial and spectral information either simultaneously or independently.

Finally, a unified multispectral synthesis approach is introduced that extends the statistical constraint enforcement framework in the P/S method into the spectral domain. Results suggest that this new framework is extremely powerful and a promising new direction. It is however, quite different than the approach taken thus far in this work. No longer is the concept of a generalized Markov Random Field model needed to extend monochrome synthesis techniques to handle textures with arbitrary spectral dimension. Gone is the important notion of abstracting the spectral content of an texture. Since this new approach lies outside the central modelling principles presented in this work it is included as an extension rather than being given an integral focus.

9.1 Hole-Filling

The hole-filling problem is most often applied to photo restoration problems. In a slightly different application the primary objective is to deliberately remove objects from a scene and replace them with meaningful information extracted from the background surrounding the removed object. In multispectral imagining this is a particularly useful operation and can be accomodated quite easily with the synthesis techniques presented in this work.

In the hole-filling problem, also referred to as constrained synthesis, the idea is to extend the boundary of the missing region inward to fill in the missing information. The formal technique of *inpainting* is particularly useful when this task is approached by a human. A human simply projects lines and curves through the missing region and then fills in the remaining area with appropriate colors. Attempts at mimicking this operation automatically have met with mixed results.

Using finite difference and reaction-diffusion methods [5, 2], a number of researchers have been able to restore scratches or thin occlusions with a high degree of accuracy. Unfortunately these impressive results seem to be limited to cases where the boundary to be inpainted is relatively thin as compared to the frequency content of the image. Basically this happens because if the scratches are thin enough, the surrounding boundary will likely lie in a homogeneous region in the image. Since the information at the boundary will be the same as the missing information the inpainting operation is relatively straight forward.

Contrast this to the situation where the region to be filled is quite large. In this case, the boundary pixels may not give enough information to project very far into the missing region. The problem is that these inpainting techniques use gradient projections based solely on a few pixels from the edge of the boundary. Given that the region to be inpainted is thin, the information from these pixels will likely not change much spatially. This is exactly analogous to a Taylor series approximation of a function about a point. In this case, the more complex the function, the smaller the radius of convergence and the higher the order needs to be to achieve small error. To combat this problem texture synthesis techniques can be used.

Contrary to a fixed order approximation based on the boundary of a scratch, texture synthesis techniques can use information from anywhere in the image to fill in the missing region. In addition, a separate image may be provided to give more data from which to estimate the statistics necessary to

fill in the missing region. Interestingly enough the concept of texture becomes somewhat more general in this case and is based more on the concept of scale than the actual characteristics of the image containing a scratch.

To elaborate, it's quite conceivable that an image that contains a scratch may not fit the criteria of being stationary ergodic. For instance, the image might be of a human face. As previously mentioned, the synthesis techniques presented in this work cannot handle this type of image. However, in the region of the scratch, the image characteristics may very well be stationary. If the image itself is viewed as a mosaic of stationary homogenous texture regions then texture synthesis can be applied locally. Not to be confused by the earlier discussion, it is reasonable to assume that *both* gradient based and texture based hole-filling techniques will to work better when the boundary of the missing region is homogeneous. The difference is that the gradient projection techniques require that the homogeneous region be somewhat simple whereas texture synthesis techniques can conceivably function equally well regardless of the complexity of the homogeneous region. Again, this is due to the fact that gradient based in-painting only considers pixels on the margin and can't reliably capture spatial correlation that is highly variable over large distances.

The synthesis based hole-filling procedure requires a masked image delineating the spatial extent of the region to be filled. Then, the image is projected into an orthogonal color space. Using the nearest neighbor resampling method, similar regions are identified elsewhere in the image. The set of such regions defines a non-parametric MRF that is randomly sampled to produce a new pixel in the missing region. The missing region is filled in this manner proceeding in an inwardly spiralling fashion from the margin. Given a separate sample image, the set of nearest neighbors can be extended to include this image. After the region is filled, the resulting image is back-projected and the result is a new image without a hole.

9.2 Expanding Spectral Coverage

The problem of expanding the spectral content of an image, whether the image is a texture or otherwise, can be addressed using statistical constraint enforcement under the assumption that the spectral data is jointly Gaussian. What's further, contrary to a re-sampling approach based on nearest neighbors, constraint enforcement doesn't introduce a dimensionality problem when using multiple bands to constrain the spectral content of the re-

(a) (b)

Figure 59: Hole filling illustrated.

sulting expanded spectral image.

The technique is simple. First, mean and covariance statistics are estimated from a spectral database of ground truth observations. These statistics serve as target constraints on a new expanded spectral image. These constraints are enforced in an iterative fashion beginning with an *augmented noise image.*

The augmented noise image contains the same number of bands as the desired spectral coverage, i.e. it has the same dimensionality as the ground truth data. It's referred to as *augmented* because scaled noise images are added to an existing image so that the resulting image cube is of the required spectral dimensionality. These noise bands are added in such a way as to pair the bands in the original image with an appropriate band in the ground truth data. This technique can operate on existing images that may be grayscale, color or multispectral. The image doesn't have to be texture either, this is a general methodology that can be applied to any image.

The idea is that the spectra in the augmented noise image must be changed subject to the constraints estimated from the ground truth *and the spatial information provided by the original image.* Since the statistical constraints are the spectral mean and covariance, this can be accomplished by simple linear transformations performed on the spectra of the augmented image.

In order to adjust covariance statistics we seek a matrix $\boldsymbol{M}$ that can enforce a particular covariance over a set of data.

$$
\begin{aligned}
C_T &= E\{MXX^TM^T\} \\
C_T &= MC_XM^T
\end{aligned}
$$

So M needs to normalize the covariance of X and then enforce a new covariance:

$$
\begin{aligned}
X' &= MX \\
X' &= V_T D_T^{1/2} D^{1/2} V^T X
\end{aligned}
$$

The term, $VD^{1/2}$ is simply the principle components analysis (PCA) transform that normalizes the covariance of X. In other words, V is a set of eigenvectors and D is a diagonal matrix of singular values. The subscript T denotes the term is derived from the target data.

After performing a covariance enforcement the augmented image is re-meaned. Unfortunately, the information in the original spatial bands has been invariably corrupted and it is doubtful that the augmented image has good spatial correlation in any of the noise bands. However, each noise band should hopefully display some spatial correlation. With any luck at all, the current results can be improved in an iterative framework.

This iterative framework repeatedly enforces the statistical constraints on the augmented noise image. At each step in the iteration, the spatial bands in the augmented noise image are replaced with the original bands. In effect, this operation re-enforces the spatial constraints. So in effect we have fixed the error introduced by the mean and covariance adjustment. This process continues until convergence to an expanded HIS image with "good" spatial information in each band.

Experiments have been performed using this technique. Unfortunately, convergence seems to be an issue. Empirical results suggest the algorithm tends to oscillate rather than converge absolutely. Therefore, an appropriate stopping condition is needed. One suggestion is to stop when the average autocorrelation of original bands best matches the autocorrelation of the corresponding band in the augmented image.

The idea here is simple, drive a iterative linear operation with a combination of noise and actual image data. The operator is applied solely in the spectral domain and amounts to a simple PCA projection as previously described. Ultimately it simply relies on simultaneous enforcement of first and

second order statistics that must "pass through" certain points at selected bands. Those selected bands are obviously the bands in the original image. An interesting aspect of this process is that is should realize better performance with more bands. Consider a somewhat trivial case of moving from 224 to 225 bands. It's unlikely this last band contains any new information. This introduces the notion of continuity in the uniform convergence sense. In short, given a sufficient number of bands and provided that neighboring bands are correlated, it is possible to interpolate with small error. This technique is essentially an interpolation/extrapolation technique that fills in the missing data.

An experiment demonstrating the effectiveness of this approach is performed. Starting with a 22 band MISI image, a band is selected at random and then augmented with 21 bands off scaled noise. The spectra in the original image serves as the ground truth data from which the necessary statistics are estimated. At first glance this might seem overly simple. Albeit this is an idealized case, it is by no means trivial. The actual spatial correlation in the full 22 band MISI image is never known. The ground truth is simply a set of column vectors that are used to estimate a mean vector and covariance matrix. This is done one time, and the spectra and MISI image are effectively discarded.

Using the single band augmented with scaled noise, 21 new bands are created to replace the noise. The results are transformed into a 3D color space using k-means clustering. This simply facilitates viewing the texture. One important thing to note is that each image, regardless of iteration, has exactly the same spectral mean and covariances as the ground truth. Given the spectral data is joint Gaussian, then the resulting augmented spectral image has the exact spectral statistics as the ground truth, i.e. Gaussian correlation is unique up to order two.

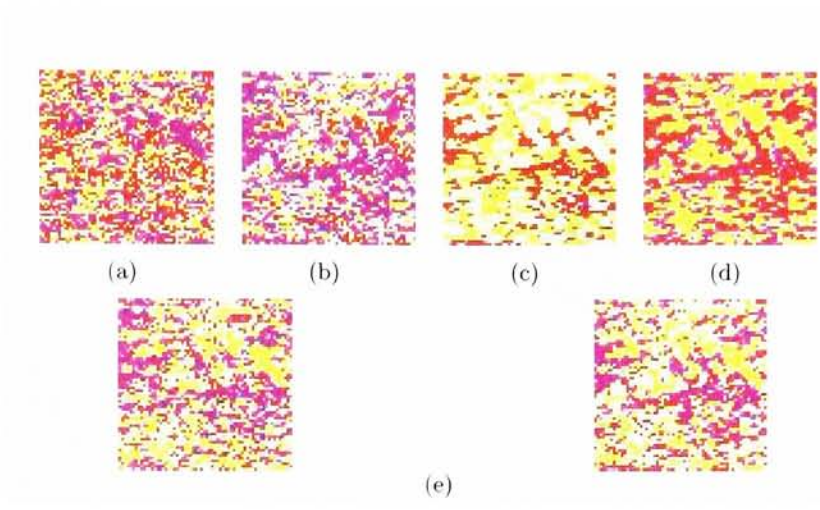


Figure 60: Spectral expansion showing intermediate results.(a-d) are successive iterations, final result is on left of subfigure e, original is on right

An additional experiment using an RGB image is performed to demonstrate the effectiveness of the algorithm in low spectral dimension. By operating in RGB space, the resulting image can be easily compared with the original from which it is derived. In this experiment a single channel is taken from a sample texture. Two additional noise bands are added to this single channel to create an RGB synthetic image. Next, the covariance enforcement technique previously described is applied to this noise augmented image. Additionally, some local smoothing is introduced in combination with a final histogram match of each band.

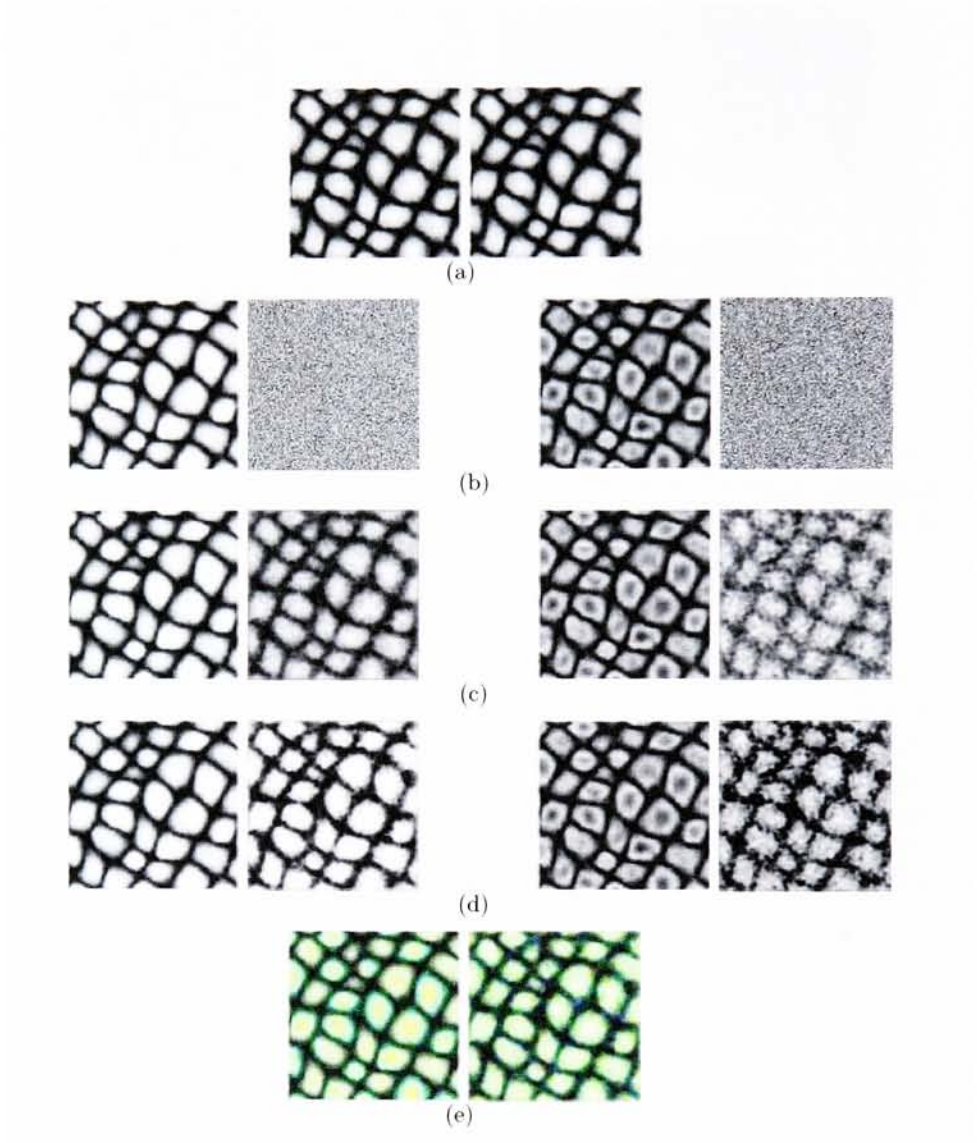


Figure 61: Spectral expansion showing intermediate results. (a) is the original band, (b)-(d) show successive iterations with local smoothing. The final results are shown in figure (e) (expanded image is on right)

9.3 Multispectral Synthesis Using Low-Order Constraint Enforcement

Encouraged by the results obtained when attempting to expand spectral coverage, a new multispectral synthesis technique is introduced. The technique is based on the P/S method to again provide a monochrome principle band using low-order constraint enforcement in a complex wavelet framework. The difference is that rather than using a resampling approach to fill in the residual spectra, another low-order constraint enforcement step is applied to the spectral content of the synthetic texture. This last step is equivalent to the approach taken to expand the spectral content described in the previous section.

The advantages are that a true unified model is obtained for the entire multispectral synthesis process. This unified model is a direct extension of the P/S method as applied to monochrome texture synthesis. Using low-order statistical constraint enforcement, the synthesis process is no longer bound to an essentially ad hoc graphics based algorithm, i.e. resampling. Although resampling has been shown to be very effective and also very fast, it has a tendency to simply copy the information in a sample texture without truly abstracting the data. The advantages of a compact abstract representation is that a number of analytic options are available for working with and analyzing the synthesis process. Using the rich calculus of linear matrix and linear system theory it is quite easy to modify the statistical characteristics of an ensemble of texture images.

This is particularly useful for adjusting the spectral covariance of a texture, or introducing a new spatial correlation to an existing texture. In general, by representing the spatial and spectral content of a multispectral texture as a set of compact low-order statistics that are easily related to average power, power spectral density, autocorrelation, etc., it is quite easy to analyze textures as random signals. This is particularly useful for signal processing applications like target detection, image segmentation, etc. that may benefit greatly by having a meaningful description of the texture information in the "language" used to design such systems.

The technique is simple. The first step is to decorrelate the bands of an input texture. Next, the P/S method is used to synthesize the principle band. Using the covariance statistics of the whitened sample image, the remaining synthetic bands are created through a simple iterative constraint enforcement as described in the section on expanding spectral coverage. Then

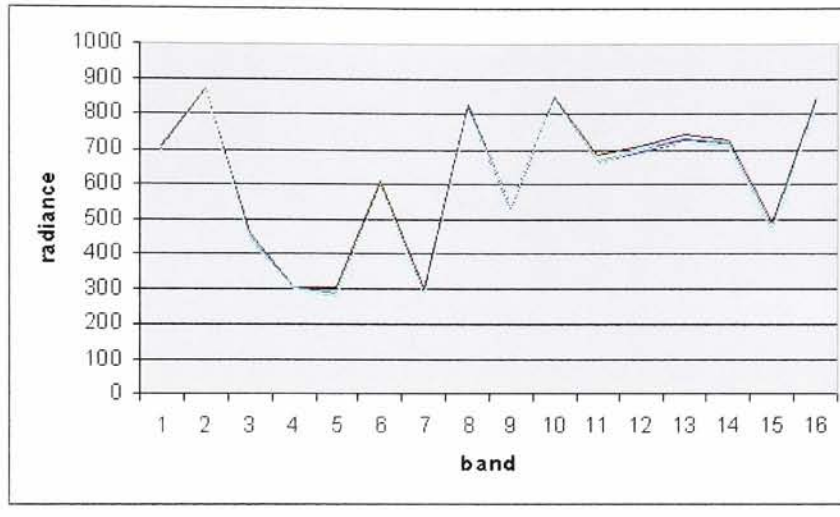
the resulting synthetic image is "colored" using the inverse transform used to decorrelate the input texture.

By operating entirely in the orthogonal color space, the spatial information remains condensed in the principle band. This principle band contains a majority of the spectral information as well and to a large degree controls the generation of the residual spectral information. When this intermediate spectral information is combined through the coloring transform, the resulting spectra is highly dependent on the information in the principle band. Hence, the overall result is controlled to a large degree by the quality of the monochrome synthesis step.

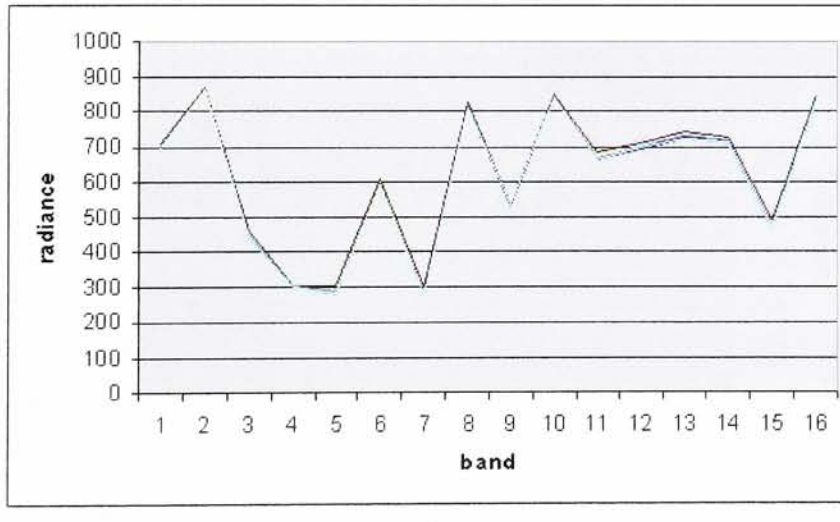
Since the monochrome synthesis step can be performed using any algorithm, the overall framework consists of essentially two orthogonal components one to create the spatial information and another to generate the spectral information. Having the flexibility to change the individual model used for each component is important, allowing for the incorporation of a number of synthesis options within a common framework. In fact the addition of this new approach fits into a common toolbox for synthesizing multispectral texture based on the view of separating the spatial and spectral processing into separate models. Although this is the original intent of this work, the spectral information is no longer abstracted.

9.3.1 Results

The results of some simple experiments using this technique are presented in Figures 62, 62, 64. The idea here is to demonstrate the potential usefulness of the algorithm, foregoing an in-depth exploration of the techniques ability to synthesize multispectral texture. For this purpose two 16 band images are synthesized. The spectra and clustered results are presented.

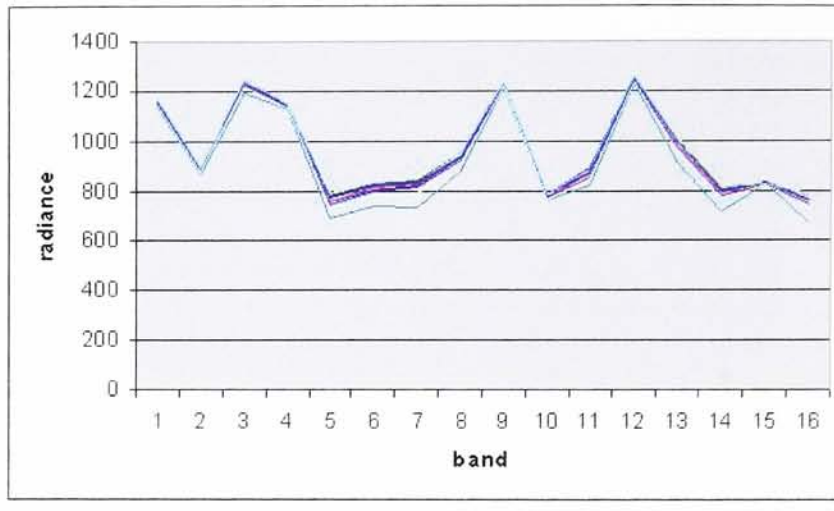


(a)

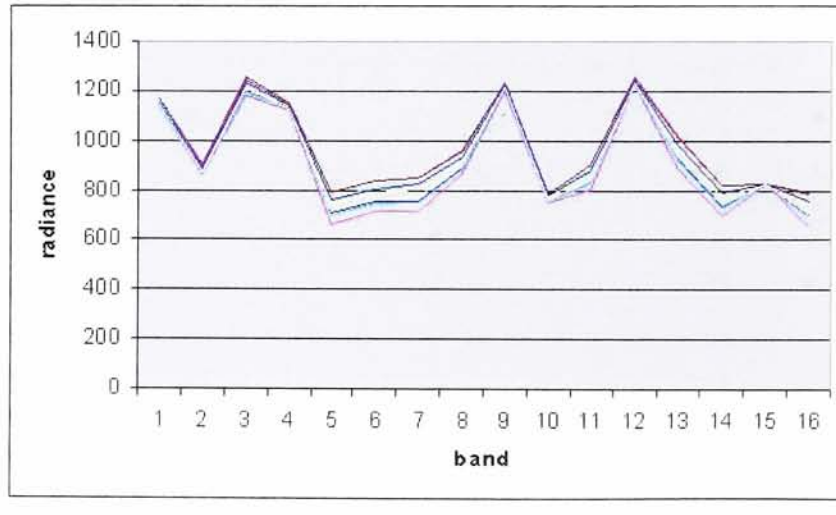


(b)

Figure 62: Spectra sampled from 16 band texture a) sample b) synthetic.



(a)



(b)

Figure 63: Spectra sampled from 16 band texture a) sample b) synthetic.

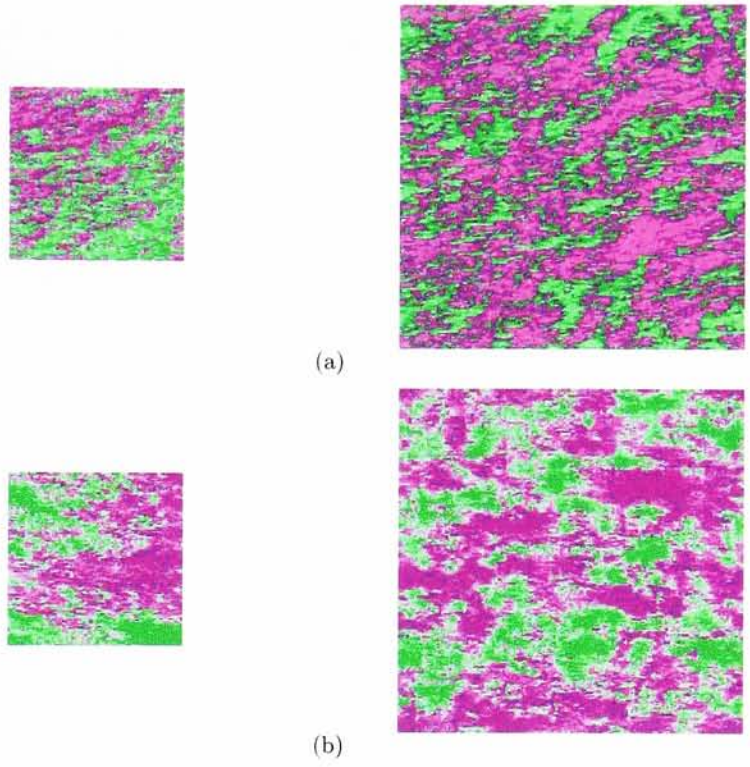


Figure 64: Multispectral synthesis results using covariance enforcement.

9.3.2 Conclusions

The results using this method appear to be quite good. Of particular note is the high degree of agreement between the spectral content of the synthetic and sample images. This is an important result since this is the first technique that generates the spectral information rather than simply extract it from a sample texture.

An interesting aspect of this algorithm is that it is fundamentally different than the approach used to expand spectral coverage. The difference is that this technique operates in an orthogonal color space where the principle band is essentially known. The result is that while the covariance constraint enforcement in the orthogonal color space still tends to oscillate rather than

converge absolutely, the coloring transform tends to suppress the variation in the residual bands. In other words, even thought the intermediate constraint enforcement steps display some undesirable variance, the principle band is the dominant source of information in the coloring transform, so this variation becomes negligible.

The results shown here only contain 16 bands. It is likely that textures with more bands may prove more difficult to synthesize. This shortcoming can be addressed fairly easily by synthesizing more than one spatial band of texture. Using the P/S technique, intermediate results can be readily modified to better match the known statistical characteristics of an input texture. This is a powerful result allowing for a progressive building up of spatial channels in the primary texture bands. In this approach the principle band is synthesized and a two channel texture is created by enforcing the covariance of the first two principle bands. This will have the effect of correlating the principle features in the texture that will likely appear in multiple bands, e.g. edges and lines. Next, these two bands are feed back into separate P/S synthesis steps and the covariance statistics can again be enforced. Incidentally, this is precisely the way the P/S method works. In this case instead of operating on adjacent bands in a wavelet representation, the algorithm is applied to adjacent bands in the spectral domain. This process can continue by adding more and more bands until enough of the variance in the spectral information has been accounted for.

Recall that during the analysis of the multispectral synthesis results the concept of a wide sense stationary process was introduced. In the discussion that followed the statement was made that the P/S method can enforce WSS statistics on a single band but that this result couldn't be extended to multispectral texture using resampling. The framework presented here can in fact extend this result. Hence, it represents another important tool in the multispectral synthesis toolbox.

10 Future Work

The limitations and weaknesses of each of the algorithms presented in this work suggest directions for future work. There are a number of methods that might be extremely beneficial for improving algorithm performance. Resampling stability is an important issue to both quilting and P/S. Improving stability would significantly improve the synthesis results of each approach.

As discussed, resampling suffers from two types of instability. Note that this discussion applies to monochrome as well as multi-channel texture. The first type of instability arises when the resampling procedure becomes completely random. In other words, the synthetic texture lacks any spatial correlation. Before discussing possible solutions to this problem, consider first the remaining way quilting, or resampling in general, can become unstable. This second type of instability is realized when the search process can't find sufficient numbers of similar neighborhoods. In other words, the search gets stuck in one area of the search space and the entire process becomes deterministic. This is the worst possible case since the resulting texture will bear no resemblance to the input texture in either the spatial or spectral domains. Since textures are being modelled as stationary processes in this work, given that this is a valid assumption, the statistics over the local spatial neighborhood inherent to all resampling procedures must display stationary statistics. Even in cases where the input texture is not stationary, low order statistics such as mean and variance will likely not vary much over a local neighborhood. Now consider that a degenerate deterministic search process will very quickly violate this stationarity. Using the stationarity criterion it should be possible to determine when the search has become degenerate. Flagging such instances will initialize an iterative process whereby degenerate points are filled in with noise and flagged for consideration in a subsequent pass.

In order to improve the stability of the quilting algorithm for the other two cases, namely when the search is producing poor results but has not yet become deterministic, the problem becomes more difficult. One possible solution is to create a low order dependency between the determination of the current and subsequent quilting blocks. In this case a set of possible texture regions are considered at each step. Next, given this small set, the block is chosen that maximizes the similarity of the boundary at the next step. In general this procedure could be extended to multiple steps in the future. This way synthesis steps are made non-causal which may be better in some cases.

Another idea that might improve the synthesis quality of the quilting method is based on in-painting. To reiterate, in-painting is similar to the way in which an artist may restore an invalid region in an image. Namely, information from the boundary of an invalid region of an image is propagated inward until the invalid region is replaced with information constrained by the boundary of the region under consideration. In the context of quilting, in-painting could be used in place of the min-cut determination. A number

of techniques for performing in-painting are presented in the literature [5, 2]. In-painting has the potential to eliminate the edge discontinuities observed with the current quilting methods.

It has been mentioned several times that the P/S method is unique in that it can guarantee enforcement of low-order image statistics in monochrome textures. Extending this result to multi-channel texture could have potential benefits. The principle work described in the section on extensions is an important first step. Additional benefits may be realized if the concept of a wavelet basis representation is extended into the spectral domain as well. This is relatively straightforward and may prove to be an important and useful generalization of the P/S method.

11 Appendix A. Wavelets

Wavelets are a relatively recent invention, used for localizing frequency information in a signal. A Fourier transform can break a signal down into a series of coefficients representing the spectral content of a signal. Unfortunately the Fourier transform doesn't give any information as to when these signals appear. One way to try and overcome this problem is to use a windowed Fourier Transform. A windowed Fourier Transform uses a windowing function to restrict the basis functions to have finite support. As a result, the Fourier coefficients represent the frequency content at a specific location in the signal, specifically where the windowed basis functions are not zero.

By shifting the location of the window over the time/space axis in a 1D signal, a series of coefficients are obtained. It is important to note that the new representation now has two dimensions corresponding to the frequency information from the windowed Fourier transform localized in time/space.

$$F(p,q) = \int f(t) e_{p,q}(t) dt = \int f(t) q(t-p) e^{2\pi q t} dt$$

where p, q represent the new dimensions of location and frequency respectively. One of the principle properties which undermines the usefulness of windowed Fourier transforms is the *uncertainty principle*. This principle states one cannot resolve frequency and space information to an arbitrary degree of accuracy simultaneously. In other words to accurately determine the frequency information in a signal one must analyze a large portion of the

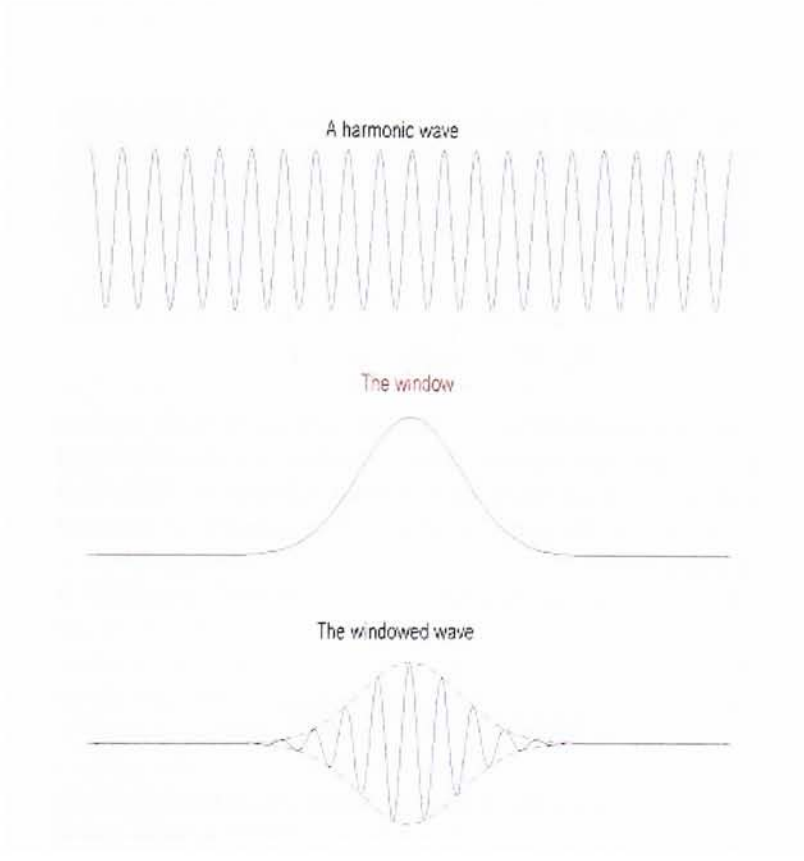


Figure 65: The windowed Fourier transform [1].

signal in the spatial domain. Consider the extreme case where an window of infinite support is used. In this case we have perfect frequency resolution but no idea of where they occur in the signal. In other words, we are back to the original Fourier transform. The practical significance is that windowed transforms are good at resolving high frequency information at specific locations in the spatial domain, but low frequency information is difficult to localize. In order to balance this problem and arrive at a compromise we generalize the concept of a windowed Fourier transforms to the wavelet transform.

Basically in generalizing the windowed Fourier transform to the wavelet transform the windowed complex sinusoids representing the Fourier basis are replaced with a function that includes scaling and shifting parameters to represent an arbitrary basis:

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{S}}\Psi_{s,\tau}(\frac{t-\tau}{s})$$

where the s,t parameters are in the dimensions of scale and time/space respectively. Clearly the wavelet basis is simply a normalized shifted copy of some function, with an additional scaling parameter. This function has a special name and is referred to as the *mother wavelet* or alternatively, the *analyzing wavelet.*

The resulting continuous wavelet transform (CWT) is defined as:

$$\gamma(s,\tau) = \int_{-\infty}^{+\infty} f(t)\Psi^*_{s,\tau}(t)dt$$

where $\gamma(s,\tau)$ represents the *wavelet coefficients*, $f(t)$ is the 1D signal, and $\Psi^*_{s,\tau}(t)$ is the *analyzing wavelet.* The fact the CWT is formulated with an arbitrary basis is what gives wavelet analysis its analyzing power. Different analyzing wavelets can be chosen for specific applications. Of course not just any wavelet can be chosen.

There are two properties of admissibility and regularity that a wavelet must adhere to. The first is that an analyzing wavelet must be have finite energy in its spectrum.

$$\int \frac{|\Psi(s,\tau)|^2}{|\omega|}d\omega < +\infty \qquad (1)$$

The result is that wavelets are band limited. This is the so- called admissibility condition. An interesting aspect of this condition implies that the average value of the wavelet must be zero, ie it is oscillatory, meaning it must be a wave (otherwise the integral will blow up at zero). Second, is the regularity condition which states the energy in the wavelets frequency response should fall off quickly as the frequency decreases. As mentioned, the average value must be zero, hence the first order moment in a Taylor series expansion of $\gamma(s,\tau)$ is guaranteed to vanish. By forcing the first n moments in the expansion to vanish the resulting wavelet will display smoothness and locality in both time and space. The smoothness is important so that its spectrum is compact. This compactness is important so that the wavelet acts like a band pass filter in the frequency domain. In the time/space the localization property means it is a wavelet instead of a wave.

Now that the continuous wavelet transform has been introduced, the next step is to discuss some of its limitations in the current form. One of the important properties of the Fourier transform is that its basis functions are orthogonal and span the entire subspace of frequency components. This means the transform is invertible, a nice property that allows us to "undo" the transform. Currently nothing has been stated about the orthogonality of a wavelet basis used to form a CWT. This is an important issue since in general, a wavelet basis is not orthogonal. To illustrate why, consider translating the analyzing wavelet at infinitesmily small increments using a wavelet that is not orthogonal to itself when translated by arbitrary shifts e.g. a single period of a square sine wave. The resulting coefficients will display infinite energy regardless of the energy of the original signal, provided it is not zero. Clearly we will not be able to invert this transform. The problem lies in the redundancy of the representation. In this case we would say that the wavelet coefficients were infinitely redundant.

In order to create an orthogonal basis or least one that is compact enough to be useful, it is necessary to reduce this redundancy. This can be achieved by simply controlling both the time/space and scale intervals at which we shift and scale the analyzing wavelet. This is precisely what the discrete wavelet transform does. The discrete wavelet is defined as:

$$f(t) = \sum_{j,n\in\mathbb{R}} \langle f, \Psi_{j,n} \rangle \, \Psi^*_{j,n}$$

Notice that the discrete wavelet transform still operates on continuous

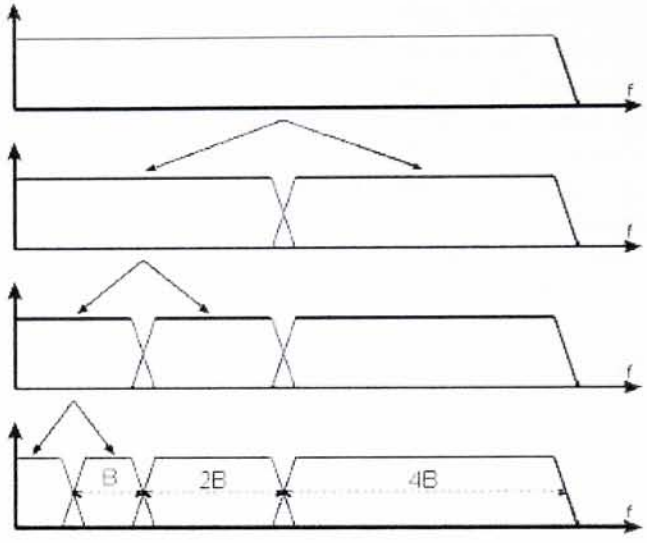


Figure 66: Subband coding [1].

signals. We simply discretize the j, n i.e. s, t parameters. The result of this operation is that we produce periodic band pass versions of the original signal in the frequency domain. This is essentially what subband coding does and is illustrated here:

Beginning with the band limited interval containing the original signal, we iteratively separate the high and low frequency information using appropriately scaled filters. Generally these filters split a band limited region into two equal parts corresponding to a high and low pass filtering operation. During the iteration process each successive high pass filtering operation creates a band pass response to the input signal. By scaling and translating each filter the correct amount in the spatial domain, the frequency information is split into successively coarser scales with a minimum amount of overlap between subbands.

Since we are operating on a continuous signal, at some point this iterative process must be terminated leaving a portion of the low frequency informa-

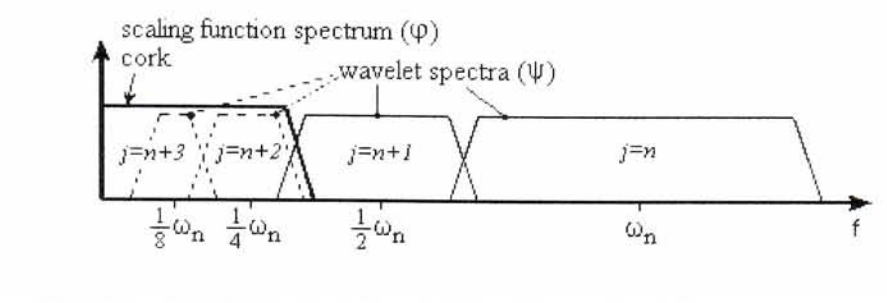


Figure 67: Illustration of the scaling function [1].

tion unaccounted for. Otherwise it would continue to operate on infinitely narrow low frequency subbands. This is analogous to a half-life in radioactive decay, you can't half something into nothing. If we simply discard this residual information the signal cannot be correctly constructed. So to compensate for this, the final iteration is replaced with the *scaling function.*

$$\varphi(t) = \sum_{j,k} \gamma(j,k) \Psi_{j,k}(t) \tag{2}$$

The scaling function can be described as a "cork" that fills the hole left over by our need to terminate the wavelet approximation. Basically the scaling function is the information left over from the original signal after the last iteration of the wavelet transform.

The resulting discrete wavelet approximation can be defined as:

$$\begin{aligned} f(t) &= \sum_{j,n\in\mathbb{R}} \langle f, \Psi_{j,n} \rangle \, \Psi^*_{j,n} \\ f(t) &= \sum_{j} \langle j, \phi_{j,n} \rangle \, \phi^*_{j,n} + \sum_{k\leq j,n\in\mathbb{R}} \langle f, \Psi_{j,n} \rangle \, \Psi^*_{j,n} \end{aligned}$$

where the low frequency is replaced with the scaling function and the remaining information is determined using the wavelet basis.

Under the right conditions the discrete wavelet transform can be inverted to achieve perfect reconstruction. When a wavelet basis can perfectly reconstruct a signal it is referred to as a *frame.* The following condition must be satisfied for a wavelet basis to form a frame.

$$A \|f\|^2 \leq \sum_{j,k} |\langle f, \Psi_{j,k} \rangle|^2 \leq B \|f\|^2$$

This condition states that the energy in the wavelet coefficients is bounded by two constants A, B. When $A == B$, we call the resulting basis is called a *tight frame*. Tight frames act like an orthonormal basis, meaning the analyzing and reconstructing wavelets are complex conjugates of each other. This is similar to what is seen when using the Fourier transform. What's further, when $A == B == 1$ the tight frame is actually an orthonormal basis.

At this point some important properties of the discrete wavelet transform have been introduced but the signals that it operates on are still continuous. What's needed is a way to perform the discrete wavelet transform on sampled signals. Fortunately this is easy to do by introducing a new way to look at the wavelet transform.

Given a sampled signal, a natural and useful way to modify the DWT is to implement the wavelet basis as a digital filter bank of finite impulse response (FIR) filters. Consider the subband coding scheme again except this time the signal is discrete. The spectrum is of course band limited and we can split the frequency domain in two (approximately) equal parts using a high and low pass filter. By iteratively applying the filters to the low pass residual and scaling the time/space parameter by a power of two, the frequency information in each band limited interval is halved. This operation leads to what is known as the *multiresolution formulation* of the wavelet transform.

In this new discrete system we can fully determine the set of possible subband coefficients. The scaling function is now a discrete set of coefficients from the previous low pass filtering operation at the next highest scale. It is natural then to refer to the *low pass filter* as the *scaling filter* since it generates the scaling function at each scale. Subsequently, the high pass filter determines the wavelet coefficients at each scale and is therefore referred to as the *wavelet filter*. At this point we no longer refer to the continuous scaling and wavelet functions of the CWT and DWT. Instead, we refer to the scaling filter h and the wavelet filter g.

The subsampling operator is used to change the scale of a digital filter by a factor of two. This means it's possible to implement a multiresolution

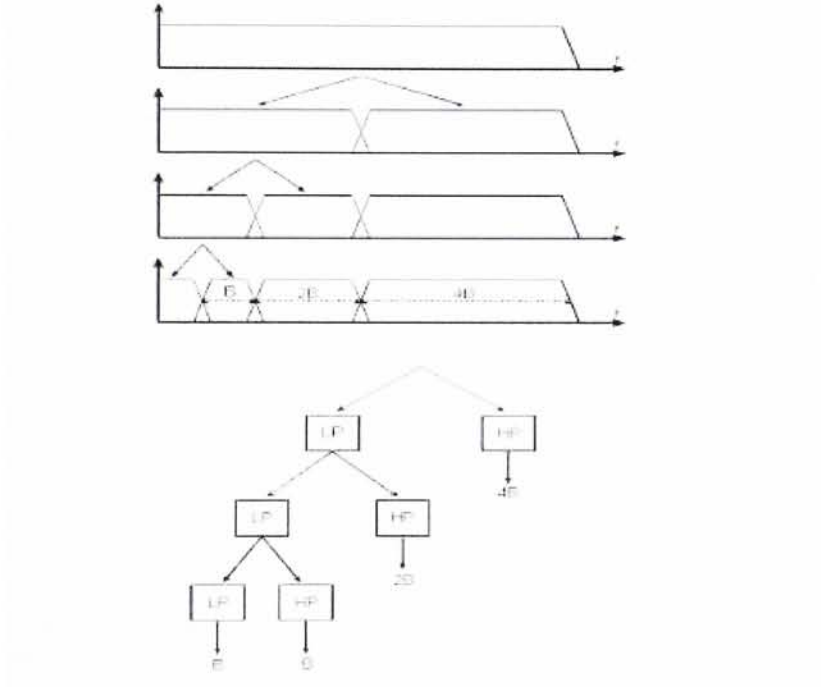


Figure 68: Illustration of the multiresolution transform in the context of subband coding [1].

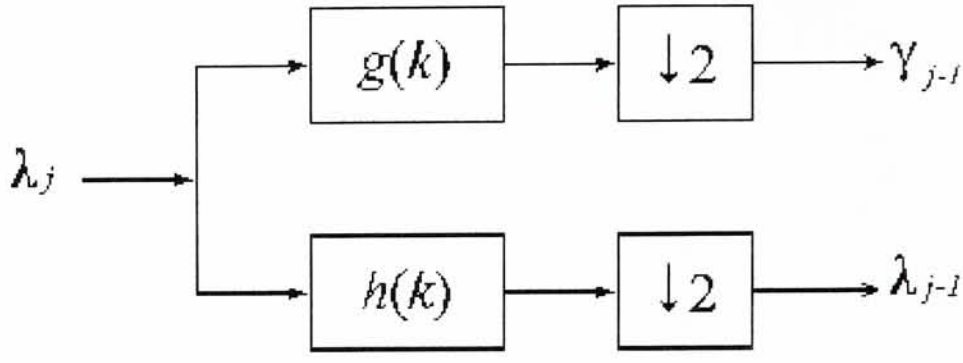


Figure 69: A single stage in the recursive application of filters g and h in order to determine the wavelet coefficient lambda. Figure also shows the *downsampling operator* [1].

transform using a single pair of digital filters by downsampling the signal by a factor of two at each iteration.

The next question is how to invert the multiresolution transform. We are still interested in the concept of a frame. Generally, any basis that forms a frame is invertible. Of particular interest are two classes of basis functions that satisfy the conditions necessary for perfect reconstruction. These two classes are orthogonal and biorthoganal filter banks. Orthogonal filter banks form orthornormal bases. A common approach for constructing orthogonal filter banks is to use halfband *quadrature mirror filters* (qmf) also known as *conjugate mirror filters* (cmf). In keeping with the general scheme outlined above, halfband qmfs form a high and low pass filter pair used to iteratively decompose a signal into equal high and low pass responses. An additional property concerning reconstruction states that the inverse operation is performed using the complex conjugate of the forward operation, meaning halfband qmfs form a tight frame with $A == B == 1$.

Here the upsampling operator works by inserting zeros for every other coefficient.

Conversely a biorthoganal system refers to a dual frame. As previously mentioned, a dual frame occurs when the forward and reverse filters are not

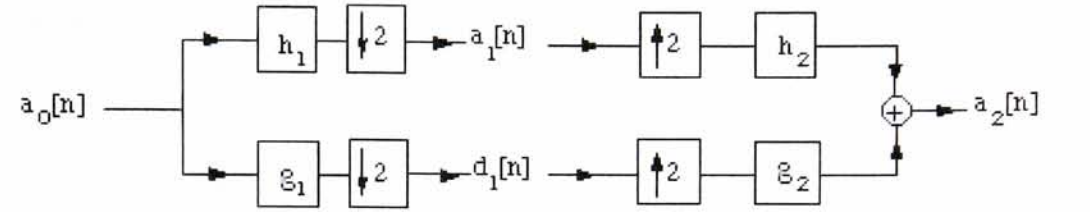


Figure 70: A single stage in the recursive application of filters g and h showing both the forward and inverse operations [1].

complex conjugates of each other. Therefore, separate forward and inverse filter banks are needed to achieve perfect reconstruction of the original signal.

The last issue concerning wavelets and multiresolution analysis has to do with extending these concepts to 2D. Like the Fourier transform, one way is to create polar separable analyzing wavelets that preserve the orthogonality of their 1D constituents. However, unlike the Fourier transform, we are no longer restricted to orthonormal bases. Some wavelet applications that aren't concerned with invertibility may benefit from using arbitrarily shaped basis functions. Consider the problem of recognizing faces in an image. In a wavelet framework a basis function capturing features in a face can be scaled arbitrarily and shifted around an image similar to the way a matched filter might work. As we will see in the upcoming examples it is even possible to add rotational information to a wavelet basis. Due to the nature of wavelets, extending them to two dimensions is heavily dependant on the application.

In order for the basis to be orthonormal the number of coeffiecients at each scale must equal half the number of samples with the other half being sent to the next level in the iteration as a new low passed version of the incoming signal.

An interesting and important result of this fact is that an overcomplete representation can be made shift invariant. This can be explained in a number of ways. The easiest is in terms of the Laplacian pyramid itself. As mentioned, the reason we retain an equal number of coefficients at each scale is that the high-pass result is not downsampled. When downsampling the high pass residual in a qmf framework the remaining coefficients are not shift invariant. Imagine the case where the high pass coefficients associated with a certain shift ended up being the same coefficients removed by the downsampling operation associated with the unshifted signal. Contrast this

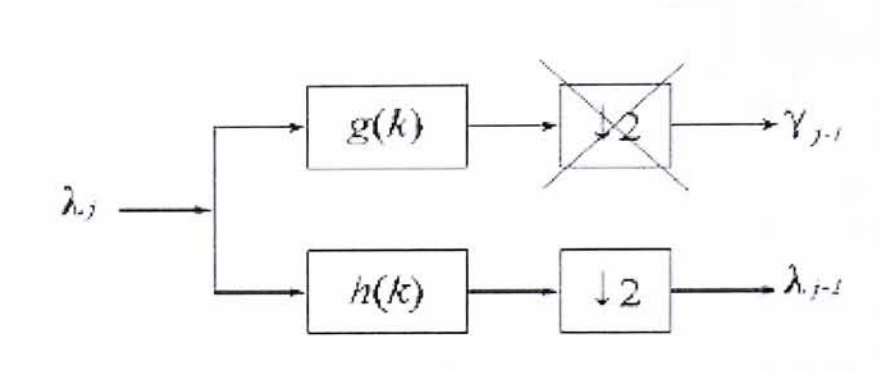


Figure 71: Overcomplete framework [1].

with the Laplacian pyramid. Shifting the original image will simply shift the result of the convolution operation. Therefore, the Laplacian pyramid is shift-invariant, a useful result. At this point one may wonder why we are not concerned about the downsampling operation on the low pass residual. The reason is that the highest frequency in the low pass filter response is half that of the input. This means half the samples are redundant and can be removed without losing any information. It is important to note that this shift invariance does not affect reconstruction. It simply affects the resulting representation's usefulness for certain applications requiring shift invariance.

Another useful property of an overcomplete basis is that they result in representations that are less sensitive to noise. Since the filters in an overcomplete basis are not shifted in increments equal to their widths the response from each filter includes some information from adjacent responses. The overall effect is to reduce the sensitivity of each filter response to impulse noise in the signal.

12 Appendix B. Color Space Transforms

We begin our discussion of color space transforms by focusing on principle components analysis (PCA). Principle components analysis has a natural

geometric interpretation with applications to the analysis of multivariate Gaussian distributions. Consider the equation $\boldsymbol{x'Ax} = \mathbf{1}$ in matrix notation, where $\boldsymbol{A}$ is symmetric, positive semi-definite. Positive semi-definite implies that all the eigenvalues of the matrix $\boldsymbol{A}$ are greater than or equal to zero. Equations of this form represent the generalized equation for an ellipse. In this case the ellipse is normalized since the equation is equal to one. To see this, consider the matrix:

$$\begin{vmatrix} \mathbf{2} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{vmatrix}$$

This matrix defines a two dimensional ellipse with minor axis 1/2 and major axis 1. This is evident if one simply multiplies through by ¡x,y¿, the result is $\mathbf{2}\boldsymbol{x}^{\mathbf{2}} + \boldsymbol{y}^{\mathbf{2}} = \mathbf{1}$. By plotting an ellipse it becomes evident that there exist a set of unique directions that are tangent to its surface. These directions are referred to as the axiis of the ellipse. Regardless of how the ellipse is rotated, it is always possible to identify these axiis. These axiis have a special property in that they define an orthogonal basis set for a vector space. The vector space in this case is formed by the row or column vectors of a symmetric positive semi-definite matrix (since we are talking about symmetric matrices the row and column spaces are equivalent).

An important property of vector spaces is the concept of a basis, ie a set of orthogonal vectors that span a space. Since the column and row vectors of a matrix define a vector space it is possible to find an orthogonal set of basis vectors that span the same space. The way this is done is by eigen analysis. Eigen analysis finds the eigenvectors of a matrix. These eigenvectors form an orthogonal basis for the vector space defined by the rows or columns of a matrix.

The eigenvectors of a matrix are defined as the set of vectors whose directions remain unchanged after multiplying by the matrix. The only thing that may change is a scaling factor referred to as the eigenvalue. The eigenvectors of a symmetric positive semi definite matrix can be interpreted as the axiis of an ellipse. Using standard techniques e.g. Cholesky decomposition, these eigenvectors can always be found for symmetric, positive semi-definite matrices. After finding the eigenvectors we can perform a *spectral decomposition* of the original matrix. This spectral decomposition involves a rotation operation by projecting the original matrix onto the orthogonal basis formed by its eigenvectors. From a geometrical point of view, this corresponds to rotating an ellipse so that the major and minor axis line up with the x and

y axis in two dimensions. This has the effect of removing the cross terms in the equation of an ellipse. In other words, the new matrix will contain only diagonal terms after projection. These diagonal terms are important and are referred to as the *singular values* of the matrix. In fact the entire procedure just described is generally referred to as *singular value decomposition* (SVD). After performing the singular value decomposition of a matrix, we are left with a new matrix of all diagonal elements. These diagonal elements correspond to the scaling on each axis of a generalized ellipse. Larger values indicate a longer axis. By sorting these values it is possible to determine which directions have the largest scaling factors. This process has a special significance in multivariate statistics.

When analyzing multivariate data it is common to form the covariance matrix from a set of observations, $\boldsymbol{C} = \boldsymbol{X}\boldsymbol{X}'$, where $\boldsymbol{C}$ is the covariance matrix and $\boldsymbol{X}$ is matrix of observations, one in each column. Covariance matrices are always symmetric positive semi-definite. Using the SVD method we know it's possible to decompose the covariance matrix into a diagonal form corresponding to the scaling factors on a generalized ellipse. This gives rise to an important concept in statistics.

The scaling factors or singular values on the diagonal of a decomposed covariance matrix represent the amount of variance a new set of random variables contribute to the overall variance in the original observations. By performing the singular value decomposition we have managed to uncover a latent set of variables that better explain the variance observed in the original data. To see how this happens consider how the data has been transformed by the SVD operation. We begin with a set of observations of random variables. After calculated the covariance matrix the off diagonals are likely to not be zero. This means that at least some of the random variables are dependant on each other. What we'd like to do is perform a transform on our observations so that we remove this dependency. This way we no longer have to consider the joint effects between random variables in our observations. We determine the appropriate transform by finding the eigenvalues of the covariance matrix. Next, we project the original data onto this new basis formed by the eigenvalues. Now if we form the covariance matrix on the transformed observations we will end up with a diagonal matrix. At this point we can sort these singular value to determine the principle components ,i.e. the directions in the original data which account for the most variance. The process of determining the eigenvectors, diagonalizing the covariance matrix, and sorting the resulting singular values is referred to as *principle*

components analysis (PCA). The name is fitting since we have identified the principle eigenvectors (components) based on their respective eigenvalues. Principle components analysis becomes very important in certain statistical applications.

Another important statistical concept follows directly from the concept of singular value decomposition. Transforming the original data so that the covariance matrix is diagonal, suggests that the variables are no longer dependant on each other. Of course a lack of dependance based on second order statistics does not guarantee that the transformed observations are independent. In fact this is only true under the special case where the original observations are multivariate Gaussian. This happens because the statistical qualities of the Gaussian distribution state that only the first and second order moments are needed to completely determine the distribution. Tying this concept back to our original discussion or ellipses, consider that the set of points that lie at a common distance from the mean of a multivariate Gaussian distribution all fall on an ellipse. This distance is generally normalized to define distance in units of standard deviation. In a multivariate setting this is the so called Mahalanobis distance. It is interesting to note then that the axis of this ellipse correspond to orthogonal random variables. When the ellipse is tilted, these orthogonal variables become latent and must be discovered using PCA. In the case of a separable Gaussian distribution the observed variables are already orthogonal ie statistically independent. So the next question is, how do we handle data that is not Gaussian? The answer is to use independent components analysis.

Independent components analysis (ICA) is related to the blind source separation problem (BSS). In blind source separation, independent sources are assumed to be mixed by a linear transform $\boldsymbol{Y} = \boldsymbol{M}\boldsymbol{X}$, and the goal is to determine the matrix $\boldsymbol{M}$ through statistical estimation methods. The end result is that $\boldsymbol{M}$ is inverted and used to minimize the mutual information between sources by rendering them statistically independent.

A particularly quick and effective method for estimating $\boldsymbol{M}$ is based on higher order statistics. It is known that a principle components transform (e.g. whitening), solves the BSS problem for linearly mixed, independent Gaussian sources. When sources are not Gaussian additional steps must be taken to remove redundancy. After a whitening operation, independent Gaussian sources display the same marginal distributions in all directions, i.e. the distribution is symmetric and the entropy in each direction is maximized meaning there is no mutual information between sources. When the data is

not Gaussian, this distribution will not be symmetric and the entropy will not be maximized. In fact there will tend to be orientations with high contrast, or high divergence from a Gaussian distribution. A Kullback-Leibler divergence metric is a good criteria for finding these orientations, but can be difficult to estimate. Generally speaking any object function for finding directions in the data that are interesting in the sense that they diverge from Gaussian might be valuable for the purposes of ICA. These objective functions are referred to as contrast functions. A particularly simple contrast function based on higher-order statistics (HOS), looks for directions of maximum kurtosis in pre-whitened data. Since a Gaussian distribution has zero kurtosis, in some sense, finding the directions of maximum kurtosis maximizes the residual information after a PCA transform.

To estimate $\boldsymbol{M}$, consider $\boldsymbol{M = USV^T}$, $\boldsymbol{M^{-1} = US^{-1}V^T}$, and $\boldsymbol{Y = MX}$, where $\boldsymbol{X = x_1, x_2, ..., x_N}$, i.e. $\boldsymbol{N}$ observations of $\boldsymbol{X}$. First calculate the covariance of $\boldsymbol{Y}$, $\boldsymbol{K_Y = E[MXX^TM^T]}$. Using singular value decomposition or spectral factorization, we can find the eigenvalue/eigenvector representation, $\boldsymbol{K_Y = USV^T}$. Assume components of $\boldsymbol{X}$ are jointly independent, then $\boldsymbol{K_Y = USSV^T = MM^T}$, and $\boldsymbol{M^{-1} = S^{1/2}U}$. Here, $\boldsymbol{M^{-1}}$ is the whitening operator and $\boldsymbol{W = M^{-1}Y}$. Notice that $\boldsymbol{M^{-1}}$ no longer contains the rotation information $\boldsymbol{V^T}$. To recover this information we need to use higher order statistics, $\boldsymbol{K^2 = E[WW^TWW^T]}$. It can be shown that $\boldsymbol{K^2V^T = \lambda V^T}$, hence to recover $\boldsymbol{V^T}$, simply decompose $\boldsymbol{K^2}$ to find its eigenvectors.

13 Appendix C. Min-Cut Determination

A minimum-cut through a matrix is defined as finding the path that minimizes some cost function. There are a number of ways to do this depending on the nature of problem, where some methods are more efficient that others. The min-cut determination needed by the quilting method presented in this work is of a special type. Namely, it is a single source network flow problem with non-negative edge weights. Problems of this type are solved very efficiently using Dijkstra's algorithm. Dijkstra's algorithm is based on another technique created by Prim for finding a minimum spanning tree (MST) of a graph structure. It is therefore useful to introduce Prim's algorithm then add the extensions needed for minimum path determination.

Before explaining the concept of a minimum spanning tree it is impor-

tant to understand some fundamental aspects of graph theory. A graph is a discrete structure consisting of a set of nodes, and a set of edges. Edges connect nodes and can be directed ie one sided $A->B$ or undirected ie bi-directional $A<->B$, where A and B are nodes. A fully connected graph is a special type of graph where each node is reachable to every other node via a set of edges in the graph. When using graphs to represent real world structures like cities on a map, it is common to associate a weight with each edge. These weights may correspond to distance, or network traffic, or electrical current etc.. Given a fully connected graph with weighted, undirected edges it is possible to determine a subset of edges that connects each node in a graph with the minimum cost while at the same time not introducing any cycles. A cycle is defined as a loop between a set of nodes. This subset of edges along with the connected nodes form a minimum spanning tree. One of the most popular methods for finding a MST is to use Prim's algorithm.

Prim's algorithm works by selecting a single starting node at random and then adding additional nodes to the MST at each stage based on the decision that minimizes the total current edge cost. In this case the decision is to choose a remaining node that is closest to any node currently added to the MST. In order to do this, the algorithm must keep track of all the distances from the remaining nodes to their closest neighbors already added to the MST. By using a data structure called a heap based priority queue, this can be done very efficiently meaning a MST can be found in $|E|log(|V|)$ time. Priority heaps use a binary tree structure to percolate upwards any changes in the weights of interior nodes. This is a very fast method of priority based sorting that always identifies the highest priority item in a queue without actually sorting the all the data. As an overview here is the pseudo code for Prim's algorithm.

```
PRIM-MST(G, s, w)
    for each vertex u in V[G]
        color[u] := WHITE       //initialize vertex u
        d[u] := infinity
        color[s] := GRAY
        d[s] := 0               //start vertex s
        ENQUEUE(PQ, s)          //discover vertex s
        p[s] := s

        while (PQ != )
            u := DEQUEUE(PQ)    //examine vertex u
            for each v in Adj[u]
                if (w(u,v) < d[v])  //examining edge (u,v)
                    d[v] := w(u,v)  //edge (u,v) relaxed
                    p[v] := u
                    if (color[v] =  WHITE)
                        ENQUEUE(PQ, v)      //discover vertex v
                        color[v] := GRAY
                    else if (color[v] =  GRAY)
                        UPDATE(PQ, v)
                    else
                        do nothing          //edge (u,v) not relaxed
            end for

            color[u] := BLACK       //finish u
        end while

    return (p, d)
```

Minimum spanning trees are very useful in real world applications. For instance, minimizing the amount of phone line needed to connect customers. Additionally, they can be used to play a role in the determination of the shortest path between nodes. To see how consider that in order for a tree to be a minimum spanning tree, it cannot be possible to connect to sets of nodes in a graph with a cheaper path without introducing a more expensive path between some other nodes that has the net effect of increasing the total edge cost. But what if we don't care about the total edge cost, just the cost of traveling from a particular node to any others. This is the essence of a

single source network flow problem. Using this logic it is quite easy to extend Prim's algorithm to find the minimum distance from a single start node to any other node in the graph. In fact we only need to make a single change to the logic in Prim's algorithm. Rather than update the distances in the priority queue with the edge weight from a node to its closest neighbor in the MST, we update using the distance between a node and the start node. This is easy to do by simply adding the distance to our closest neighbor to its distance to the starting node! Notice in the pseudo code how only a single line has changed (Figure 70).

```
DIJKSTRA-MinPath(G, s, w)
    for each vertex u in V[G]
        color[u] := WHITE       //initialize vertex u
        d[u] := infinity
        color[s] := GRAY
        d[s] := 0               //start vertex s
        ENQUEUE(PQ, s)          //discover vertex s
        p[s] := s

        while (PQ != )
            u := DEQUEUE(PQ)    //examine vertex u
            for each v in Adj[u]
                if (w(u,v) < d[v])  //examining edge (u,v)
                    d[v] := w(u,v)+d[u]  <- the only difference!
                    p[v] := u
                    if (color[v] =  WHITE)
                        ENQUEUE(PQ, v)      //discover vertex v
                        color[v] := GRAY
                    else if (color[v] =  GRAY)
                        UPDATE(PQ, v)
                    else
                        do nothing          //edge (u,v) not relaxed
            end for

            color[u] := BLACK       //finish u
        end while

    return (p, d)
```

Figure 72: Figure showing how to form a single source network from a boundary overlap.

Now that we have introduced a means for finding a minimum path between a node and any other node in a graph we need to demonstrate how to use this to solve our min-cut problem. Our min-cut problem consists of finding a path through a region that minimizes a sum over the magnitude of differences between two overlapping regions. The first step is to calculate the magnitude of overlap. This matrix defines a fully connected graph structure with edges defined as the sum of adjacent magnitudes. Next, two dummy nodes, a single start and a single end node are added to this graph structure. The effect is to sandwich the graph of interest between a single start and end node. Next, we use Dijkstra's algorithm to find the cheapest path from the start to end node. This path becomes our boundary on which to perform the min cut operation. In the case where our region consists of two overlapping rectangles, we apply this procedure twice and performing a logical and operation on the result to determine the final min cut.

14 Appendix D. KD Trees

A KD tree is a type of binary tree primarily used to solve the nearest neighbors problem. In the nearest neighbors problem the goal is to find a set of nearest neighbors corresponding to a single exemplar or query vector based on a standard distance metric, e.g. Euclidean distance. The brute force approach is to search the entire set of vectors. Running in linear time, this approach is slow. An alternative is to use a KD tree representation of the data which can be searched in approximately $log(n)$ time using a technique called orthogonal range searching. Orthogonal range searching refers to splitting a search space into regions with orthogonal boundaries. These boundaries correspond to threshold values, meaning interior or boundary points fall within

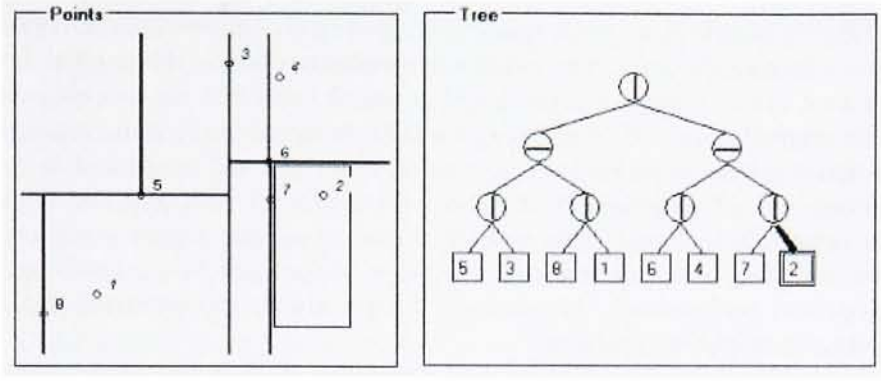


Figure 73: Figure showing KD-tree representation.

a certain range in the search space. Boundaries are determined when a KD tree is constructed. Construction follows by recursively separating a set of multidimensional data into two equal halves based on a threshold value along an axis in the search space. By rotating through the axis in the search space at each step of the recursion, closed regions are determined, each containing exactly a single data point. For instance, in 2 dimensions a KD tree partitions a search space into rectangular regions where the boundaries of each rectangle act as thresholds. Finding a nearest neighbor then becomes a matter of comparing a query vector's components against a hierarchical set of boundaries until the nearest point is found.

An important distinction between this and other methods such as tree structured vector quantization (tsvq), is that a KD tree can guarantee the single closest match is found. Tree structured vector quantization is popular in the literature and is often suggested as means of solving the nearest neighbor problem. Tree structured vector quantization imposes a binary tree structure on the code book generated by a vector quantization (vq) step. Since vector quantization is a compression tool, this code book contains a set that vectors used to represent an original data set at a reduced bit rate. In other words, the code book generally has fewer vectors than the original data set hence the ability to compress the data. When using vq the code book usually consists of a set of centroids representing the original data. These centroids are most often determined using a k-means clustering technique where k is predetermined based on the desired compression factor. In

essence, vq quantizes a vector space by replacing continuous convex regions in space by a single point. Obviously since we have replaced the original data with a set of centroids, a nearest neighbor search can't find the nearest point in the original data. Of course under a nearest neighbors formulation compression is not the main goal so it is possible to include all the original data in the code book. Unfortunately, it's still not possible to guarantee the closest data point will be found. The problem is that tsvq uses a greedy approach when searching a binary tree representation of the code book, resulting in sub-optimal performance. In summary, tsvq is always an approximation to the nearest neighbors problem.

As already mentioned, KD tree based nearest neighbor searching is exact, or rather can be made exact. In certain situations an approximate nearest neighbor search may be fine. Since the partitions in our KD tree are orthogonal it is a relatively simple matter to guarantee that the closest data point is ultimately found. This is done using a bound-overlap-ball when searching interior nodes, and a ball-within-bounds test to determine if the proper point has been found. The bound-overlap-ball is used to test if the closest point in one partition is closer to the query vector than the closest point in the alternative partition. This allows us to consider the alternative partition. Once a partition is chosen the closest point in that partition defines the radius around our query vector called the bound-overlap-ball. Once a terminal node is discovered, a ball-within-bounds test determines whether or not the latest bound-overlap-ball is entirely contained in the final partition. If it is, then the nearest data point is found, if not then backtrack and continue searching.

Given that we can determine the true nearest neighbor using a KD tree, it is a simple matter to generalize this search to discover multiple neighbors. In this case a very efficient solution can be found to a k-nearest neighbors (kNN) determination. The kNN technique is a useful tool for finding a non-parametric estimate of a probability density function (pdf). Under a kNN formulation, a pdf is estimated based on the k nearest points in a metric space. These nearest points define the local density estimate. Given a set of neighbors a pdf can be implicitly defined by sampling directly from the set of neighbors, or smoothing methods such as Parzen windows [19] can be used to estimate a continuous pdf.

15 Appendix E. Recursive K-Means Clustering

A variant of the k-means clustering algorithm is used to cluster the spectra of textures with high spectral dimension. This particular variant uses a unique approach to handle *dead clusters* i.e. clusters with no samples. Specifically, the algorithm attempts to maintain a user specified minimum representation for each cluster by splitting the largest cluster using a recursive k-means operation.

Unsupervised learning, i.e. clustering, is very useful for performing classification. In the absence of class labels, clustering techniques provide a means to learn information from a dataset. This information may be an induced classifier, an associative map, or augmented training data where the labels themselves are learned.

The k-means algorithm is a particularly simple clustering technique. The method requires the user to specify the number of clusters. Initial seeds are usually chosen randomly from the set of data to be clustered, but may be user specified or chosen to satisfy some geometric constraint. Next, a distance metric is needed and is generally chosen to be Euclidean distance. Then, the algorithm proceeds in an iterative fashion alternately assigning the sample patterns to the current cluster centroids, and then re-computing these centroids based on the assigned patterns. An additional stopping criteria is usually employed in case the algorithm doesn't converge to a stable configuration.

One of the main problems with the k-Means clustering algorithm is the so-called *dead cluster*. In other words, a cluster that doesn't have any associated patterns. In order to overcome this problem a number of approaches are possible. One method is to re-seed the dead cluster. Alternatively, patterns can be removed from other clusters to form a new centroid. In this work, a different approach is taken.

Whenever a dead cluster is encountered, the largest cluster is split into two partitions using a recursive k-means step. The centroids of the split cluster are then used to replace the smallest and largest cluster centroids. During the recursive split, the algorithm attempts to find a equal partition by iteratively balancing the clusters by shifting patterns back and forth at random. Since it is generally easier to balance fewer classes, the splitting process usually finds balanced clusters relatively quickly. To ensure efficiency,

the splitting k-means step is limited to a few iterations, usually 3 to 5.

This algorithm has been found to be particularly useful for finding internally consistent clusters when the optimal number of clusters is known a priori. In fact,experiments show that near perfect clustering can be achieved by adjusting the minimum cluster size.

References

[1] *A Really Friendly Guide to Wavelets.* http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html#section1

[2] Ballester, C., Bertalmo, M., Caselles, V., Sapiro, G., and Verdera, J. *Filling-In by Joint Interpolation of Vector Fields and Gray Levels.* IEEE Trans. Image Processing, August 2001.

[3] Barnsley, M. F., A. Jacquin, L. Reuter, and A. D. Sloan. *Harnessing chaos for image synthesis.* Computer Graphics, SIGGRAPH 1988 Conference Proceedings, 1988.

[4] Barnsley, M. F. *Fractals Everywhere.* Academic Press, San Diego, 1988.

[5] Bertalmo, M., Sapiro, G., V. Caselles, V., and Ballester, C. *Image Inpainting.* Proceedings of SIGGRAPH 2000, New Orleans, USA, July 2000.

[6] Besag, J. E. *On the statistical analysis of dirty pictures (with discussion).* Journal of the Royal Statistical Society B, vol. 48, no. 3, pp. 259–302, 1986.

[7] Botchko V., Auvinen J., Mkinen O., Klviinen H., and Parkkinen J. *Parametric Multispectral Texture Modeling.* 11th Scandinavian Conference on Image Analysis. SCIA'99, Kangerlussuaq, Greenland, June 7-11, 1999, Vol. II, pp. 763-769.

[8] Brodatz, P. *A Photographic Album fo Artists and Designers.* Dover, New York, 1996.

[9] Cadzow, J.A., Wilkes, D.M., Peters, R.A., II, and Li, X. *Image texture synthesis-by-analysis using moving-average models.* IEEE Trans on Aerospace and Electrical Systems, 29(4):1110–1122.

[10] Cardoso, J.F. *Source separation using higher order moments.* In: Proc. ICASSP, Glasgow, Scotland, 1989, pp. 2109-2112.

[11] Cormen. Thomas H., Leiserson, Charles E., Rivest, Ronald L. *Introduction to Algorithms.* MIT Press (1990), pages 527-531.

[12] Cross, G.R. and Jain, A. K. *Markov random field texture models.* IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI5, No. 1, pp. 2539, Jan. 1983. 22.

[13] De Bonet, J.S., Viola, P. *A non-parametric multi-scale statistical model for natural images.* In Adv. in Neural Info Processing. Vol. 9. MIT Press.

[14] De Bonet, J.S. *Multiresolution sampling procedure for analysis and synthesis of texture images.* Computer Graphics Proceedings, pp. 361-368, 1997.

[15] Deguchi, Morishita *Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques.* IEEE Transactions on Com- puters, C-27, no.8, 739/745 (1978)

[16] De Ridder, D., Kittler, J., and Duin, R.P.W. *Probabilistic pca and ica subspace mixture models for image segmentation.* In M. Mirmehdi and B. Thomas, editors, British Machine Vision Conference, pages 112-121, 2000.

[17] DIRSIG http://www.cis.rit.edu/research/dirs/dirsig/dists/

[18] Dodd, N. *Multispectral Texture Synthesis Using Fractal Concepts.* IEEE Trans. Pattern Anal. Machine Intelligence (PAMI), Vol. 9, No. 5, Sept. 1987, pp. 703-707.

[19] Duda, R., R. Hart, R. *Pattern Classification and Scene Analysis.* John Wiley, 1973.

[20] Efros, Alexei A. and Freeman, William T. *Image Quilting for Texture Synthesis and Transfer.* In Proceedings of SIGGRAPH 2001, pages 341–346, 2001.

[21] Efrom, A., AND Leung, T. *Texture synthesis by nonparametric sampling.* International Conference on Computer Vision 2 (September 1999), 1033–1038.

[22] Ehrich, R. W., and J. P. Foith. *A view of texture topology and texture description.* Computer Graphics and Image Processing vol. 8, pp. 174202, 1978.

[23] Fu, K. S. *Syntactic image modeling using stochastic tree grammers.* Computer Graphics and Image Processing vol. 12, pp. 136152, 1980.

[24] Gagalowicz, A. *A new method for texture fields synthesis.* IEEE PAMI, 3(5):520-741, 1981.

[25] Garber, D.D. *Computational Models for Texture Analysis and Texture Synthesis.* PhD Thesis University of Southern California, Image Processing Institute 1981.

[26] Gaskill, Jack D. *Linear Systems, Fourier Transforms, and Optics.* New York: John Wiley and Sons, 554 pp.

[27] GENESSIS http://www.photon.com/Services/Background%20Simulations/genessis.htm

[28] Hart, P.E. *The condensed nearest neighbor rule.* IEEE Trans. on Inform. Theory, 14:515-516, 1968.

[29] Hassner, M. and Sklansky, J. *The use of Markov random fields as models of texture.* Computer Graphics and Image Processing vol. 12, pp. 357370, 1980.

[30] Heeger, D. J., AND Bergen, J. R. *Pyramid-based texture analysis/synthesis.* In Proceedings of SIGGRAPH 95 (Anaheim, California), August 6–11, 1995.

[31] Julesz, B. *Visual pattern discrimination.* IRE Trans Info Theory, IT-8, 1962.

[32] Lee, E.W. and Chae, S.I. *Fast design of reduced complexity nearest-neighbor classifiers using triangle inequality.* PAMI, 20(5):567-571, 1998.

[33] Liang,.L. Liu, C., Xu, Y., Guo, B., and Shurn, H.Y. *Real-time Texture Synthesis by Patch-based Sampling.* Technical report MSR-TR-2001-40, Micrsoft Research, March 2001.

[34] Liang, Yufeng, Simoncelli, E.P. and Lei, Zhibin. Color channels decorrelation by ICA transformation in the wavelet domain for color texture analysis and synthesis. IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 2000, Page(s):606-611 vol.1.

[35] Lu, S. Y., and Fu, K.S. *A syntactic approach to texture analysis.* Computer Graphics and Image Processing vol. 7, pp. 303330, 1978.

[36] Mandelbrot, B. B. *The Fractal Geometry of Nature.* W. H. Freeman, New York, 1982.

[37] Mastin, G.A., Watterberg, P.A., and Mareda, J.F. *Fourier Synthesis of Ocean Scenes.* IEEE CGA, March 1987, pp. 1623.

[38] Paget, R. and Longstaff, I. *Texture synthesis via a noncausal nonparametric multiscale Markov random field.* IEEE Transactions on Image Processing, 7(6):925– 931, June 1998.

[39] Portilla, J. and Simoncelli, E.P. *Texture Modeling and Synthesis Using Joint Statistics of Complex Wavelet Coefficients.* Int. J. of Computer Vision, Vol.40, No.1, Oct. 2000, pp. 49-72.

[40] Portilla, J. and Simoncelli, E.P. *A parametric texture model based on joint statistics of complex wavelet coefficients.* Int. J. Computer Vision, 40:49–71, 2000.

[41] Nene, S.A. and Nayar, S.K. *A simple algorithm for nearest neighbor search in high dimensions.* PAMI, 19(9):989-1003, 1997.

[42] Shannon, C.E. *A mathemtical theory of communication.* The Bell System Technical Journal, 27:379–423, 623–656, 1948.

[43] SSGM https://vader.nrl.navy.mil/ssgm/info/proginfo.html, October, 2002

[44] Strenzwilk, D.F., Meredith, M.P., and Federer, W.T. *A Two Dimensional ARMA Model for the Simulation of IR Backgrounds.* Communications in Statistics, Part B: Simulation and Computation, vol. 18, no. 4, pp. 15391555, 1989.

[45] Turing, Alan. *The chemical basis of morphogenesis.* Philosophical Transactions of the Royal Society (B), 237:37–72, 1952.

[46] VisTex *An online collection of visual textures.* MIT Media Laboratory, 1995. Available from http://www.white.media.mit.edu/vismod/VisionTexture/vistex.html

[47] Wei, Li-Yi and Levoy, Marc *Fast texture synthesis using tree-structured vector quantization.* In SIGGRAPH 2000, pages 479-488, 2000.

[48] Witkin, A., AND Kass, M. *Reaction-diffusion textures.* In Computer Graphics (SIGGRAPH '91 Proceedings) (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 299–308.

[49] Youla, D.C. *Generalized image restoration by the method of alternating orthogonal projections.* IEEE Trans. Circuits and Systems, 25:694–702.

[50] Zhu, S., Wu, Y., and Mumford, D. *Filters, random fields and maximum entropy(FRAME) -towards the unified theory for texture modeling.* IEEE Conf. Computer Vision and Patt Rec June 1996.

Documentation for newTex 3.0 Texture Synthesis Framework

Alex Tyrrell

25 December 2002

Contents

1 Introduction

The newTex 3.0 texture synthesis framework is a single executable for synthesizing textures with arbitrary spectral dimension. The purpose of this document is to provide the user with an overview of the system, instructions for installing the software, and some hands on examples to get started. Additional advanced material discusses the use of batch files, obtaining novel data, and information on modifying and enhancing the existing code base.

The system is designed to run on windows machines as well as Unix/Linux variants. Every effort has been made to keep the code as simple as possible to ensure platform compatibility. People with little experience installing, and compiling software will probably want to enlist the help of an experienced programmer or system administrator to help locate compilers, linkers and additional resources.

The newTex 3.0 system has a *command line* interface. This means there are no windows or dialogs to help the user select arguments and files. Although this may at first seem like a hinderance, the system is basically designed to be employed in batch mode in which case a GUI would not be helpful. Users are encouraged to experiment with program options. However, sufficient familiarity with the command line interface is needed and it is probably a good idea to try all the examples before experimenting to any large degree.

Before moving on, users should also be warned that this is inherently a prototype system and therefore may become unstable and even crash! Don't be alarmed, the code has decent error handling for obvious command

line problems but exceptions caused by extreme command line options, i.e. large/small textures, etc., may crash the system without warning. On the positive side, as a prototype system, the code is very accessible for those that are interested. In fact, a great deal of powerful functionality is currently not being used by the command line synthesis variants.

2 Installation

Assuming the installation CD is available, the user has the option of installing on a windows or Unix/Linux system. In this context the term *installation* is used loosely since the entire system is a stand-alone executable with no external dependencies or other system requirements that would require adminstration priveledges.

Regardless of the target platform, the source code is *exactly* the same. This means that the entire system is platform independent. This is of important note since a great deal of effort went into keeping things this way. Without getting into too much detail yet, the code is organized into logical pieces. On the distribution CD one sees several directories each of which contain source code.

Basically the system is separated into core functionality and 3rd party source. The core functionality is contained in the top level *newTex/src* directory. Naturally, the 3rd party source is in the other directories. Additional information concerning the source will be given later.

2.1 Windows 95/98/2000/Me/NT

The system was written on a desktop PC using the Microsoft Visual C++ (MSVC++) integrated development environment (IDE). Therefore, the windows distribution is simply a MSVC++ project. By opening the newTex.dsp file in MSVC++, the entire project can be built, run and modified quite easily.

For those who don't have access to MSVC++, it is generally possible to import the entire project into say, Borland C++ Builder. This is especially easy since there is no GUI component in the code. For all other compilers, the best option is to create a single source directory and copy all the files in the project to this directory. Then simply compile the file *texture.cpp* in

this scope. Most of the functionality is included in .h files and compilation in this manner is quite easy.

2.2 Unix/Linux

Since the code was written on a windows platform, the equivalent Unix/Linux source is included along with the necessary *Makefiles* with the MSVC++ project files removed. By including this source as a single zip file *newTex.zip*, it is particularly simple to ftp the project to a Unix box, and also facilitates command line copying from the distribution CD.

Assuming that some users may not be familiar with Unix, a Makefile is a script that compiles and links large programs that have multiple sources, external libraries etc.. Each source directory has its own Makefile that generates a set of intermediate files that will be *linked* together into the final executable. Using makefiles can be *very tricky*. Generally, they require specific compilers to be stalled on the native system. At this point the code has been successfully compiled using the *HP Sparc* and *Sun Workshop* compilers. Unfortunately, the *TIPS* library doesn't seem to be compatible with the *gcc* compiler. For those using a CIS Unix box, simply use the *CC* command. Again, for those not familiar with Unix/Linux programming, enlist the help of an experienced programmer when building the system for the first time.

After extracting the source using *unzip*, i.e. *unzip newTex.zip*, the directory *newTex* will be created. In order to build the project simply change to the *src* directory via the command *cd newTex/src*. From this directory simply type the command *make* and the executable *newTex* will be built.

3 Getting Started

After installing and building the target executable *newTex*, one can begin creating textures! Accompanying the source distribution is a set of example textures. These textures are included as both a zipped archive and an equivalent directory/file structure. These textures are logically divided into gray, color and multi-spectral textures. These example textures are derived from multiple sources including the Brodatz [2] and VisTex [5] collections as well as MISI images. There is a fairly rich set of textures in this test set that can be used to really explore the effectiveness of various algorithms.

All examples will be presented as command line statements which are

platform independent. For those unfamiliar with batch or command line processing on a windows machine, additional information is provided later in this document.

4 Examples

With the exception of the multi-spectral images, any decent graphics utility can be used to process the example textures. Of particular note are *JASC PaintShop Pro* and *Adobe Photoshop*. The availability of a good graphics program is basically mandatory since the size of each texture must be provided on the command line.

4.1 Monochrome

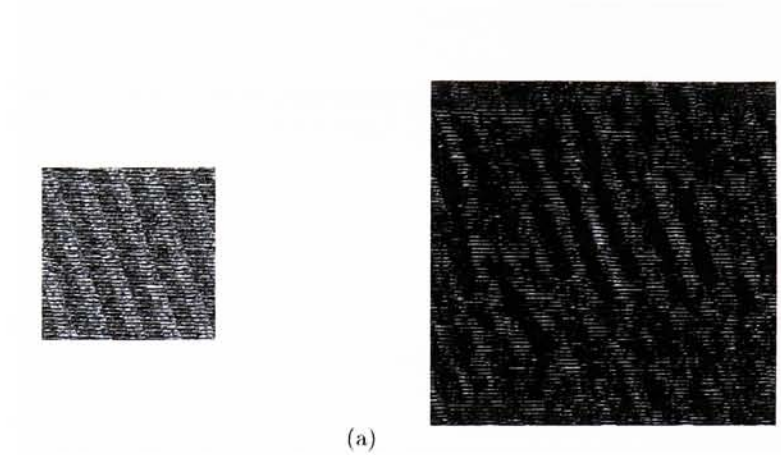
Monochrome textures must be of the file type: *.pgm*. In particular they should be type "5", i.e. binary. All the examples on the distribution CD are of this type. Basically the command line can be divided into two pieces. The first piece specifies the input/output file names, sizes and type. This information doesn't change while trying various methods. The second piece of the command line specifies the method to be used. This way trying various algorithms is relatively simple.

4.1.1 S/P Method

Our first example is to use the S/P method [4] to synthesize a 256x256 greyscale texture from a 128x128 input. The file chosen is *grey/press5128.pgm*. The command line for synthesizing this texture is:

```
newTex grey\press5128.pgm grey_out\smpress5128.pgm -c1 -ss128
-st256
```

This is perhaps the simplest command line to understand since for one thing the S/P method is the *default*, meaning there is no need to explicitly tell newTex to use it. The first statement in the argument is obviously the name of the exe. The next two arguments are the input and output files respectively. The $-c$ option specifies the channels in this case one for monochrome. Both the $-ss$ i.e. *sample size*, and $-st$ i.e. *synthetic texture size*, are set to the appropriate sizes. Note that these sizes are always *square*.



(a)

Figure 1: Synthesis results using S/P method.

This is in fact always the case!! Note that all these arguments are *required.* This means that these are the minimum arguments that must always be specified.

4.1.2 Quilting Variants

The other monochrome synthesis options pertain to the quilting synthesis variants. These include: pixel-by-pixel resampling, quilting with alpha-blend and quilting with min-cut. The arguments needed to specify these algorithms are straightforward.

```
newTex gray\press5128.pgm gray_out\abpress5128.pgm -c1 -ss128
-st256 -mquilt_ab

newTex gray\press5128.pgm gray_out\mcpress5128.pgm -c1 -ss128
-st256 -mquilt_mc

newTex gray\press5128.pgm gray_out\rspress5128.pgm -c1 -ss128
-st256 -mresample
```

The only difference this time is that the $-m$, i.e. *method* parameter is specified.

4.2 Color

In order to synthesize color texture the input file type must be *.pgm* and type 6, i.e. binary. The only difference this time is that the number of channels is now 3.

4.2.1 S/P Method

This time we try the S/P method on a color texture. Note that this is still the default method so there is no need to specify the $-m$ argument.

```
newTex color_new\Blucan.ppm color_new_out\smBlucan.ppm -c3 -ss128
-st256
```

4.2.2 Quilting Variants

The quilting variants are straightforward as well. Simply change the $-c$ option to 3.

```
newTex color_new\Blucan.ppm color_new_out\abBlucan.ppm -c3 -ss128
-st256 -mquilt_ab

newTex color_new\Blucan.ppm color_new_out\mcBlucan.ppm -c3 -ss128
-st256 -mquilt_mc

newTex color_new\Blucan.ppm color_new_out\rsBlucan.ppm -c3 -ss128
-st256 -mresample
```

4.3 Multi-Spectral

Synthesizing multi-spectral textures is straightforward. The command line for multi-spectral synthesis contains an additional argument $-t$ for *file type*. Currently only ENVI files are supported.

4.3.1 Envi Files

ENVI files must be 1) band interleaved by pixel (bip) 2) of type unsigned int (12) 3) and unix byte order (1). The program ignores the header files so don't expect the program to determine the nature of data in the file. It's still a good idea to have the header files around though since they provide information about the number of bands and texture size.

4.3.2 S/P Method With Spectral Constraint Enforcement

A new option for multi-spectral textures (or color) is the spectral constraint enforcement option. This method still uses the S/P method to synthesize a principle band and then generates the residual spectra using covariance constraint enforcement. The command line for using this option is provided here:

```
newTex color\reptile.ppm color_out\rsreptile.ppm -c3 -ss128 -st256
-msp_expand
```

4.3.3 Quilting Variants

Note the additional $-t$ argument:

```
newTex multi\1999_misi_B14.img multi_out\ab1999_misi_B14.img -c16
-ss128 -st256 -mquilt_ab -tIMG

newTex multi\1999_misi_B14.img multi_out\mc1999_misi_B14.img -c16
-ss128 -st256 -mquilt_mc -tIMG

newTex multi\1999_misi_B14.img multi_out\rs1999_misi_B14.img -c16
-ss128 -st256 -mresample -tIMG
```

5 Advanced

In order to perform experiments synthesizing multiple textures or the same texture with multiple options, it is convenient to use batch-processing. Batch-processing allows a level of automation by running an exe with multiple command line options.

5.1 Running Experiments

In order to run experiments involving multiple textures or multiple options there are some simple guidelines. The first thing is to identify the textures that you want to work with and place these in a single directory. Next, create a corresponding output directory to hold the results. By keeping this structure, it is relatively easy to browse the results, etc. and the experiments

remain separate. This is the way the data is organized in the distribution CD.

The next step is to create your batch files. This can be tricky for the novice so it is best to probably stick with the sample batch files provided on the distribution CD. It is important that the executable *newTex* is in your current *path*. This means you must explicitly copy the exe to the same directory containing your batch files, or you can add the exe to the *path* environment variable. Generally it is simplest to copy the exe to a single top level directory form which the user can run their experiments. If you decide to do this, *don't forget to recopy the exe every time the source is recompiled.*

5.2 Using Scripts

On the distribution CD there are a number of scripts that can be used on a windows or Unix platform to run experiments in *batch mode*. Use these files as an example for your own experiments. Creating these files is straightforward provided one knows a few shell commands to extract file names etc. from the file system. It is pretty simple to use a editor to perform global find and replace operations to build new batch files from the examples given.

On a windows platform batch files must have the *.bat* extension in which case simply type the name of the batch file on the command line to run the script. The file can also be clicked on from the windows file manager. On a Unix machine scripts are run by specifying the shell command *csh* or *bsh* depending on the current shell. Ask a sys admin to help you do this.

6 List of Options

A complete list of program options is provided in this section.

```
Required arguments:

input file name and path

output file name and path

-c channels (default 3)
```

```
-ss  sample size (default 128)

-st  texture size (default 128)

Optional arguments:

-i iterations S/P method (default 5)

-o pyramid orientations S/P method (default 4)

-t output type (default PGM/PPM)

-m method (default simoncelli)

-n quilting block size or resampling neighborhood size

-p nearest neighbor search precision (default 1.0)
```

7 Hints, Tricks, Debugging

There are a few known bugs in the code. Most of these bugs are not serious but the user should be aware of them. The following list is presented in order of severity.

1. Sample textures must be at least 64x64 and a power of 2, i.e. 128x128, etc. in order to use the S/P method.

2. Quilting with block sizes that are larger than 1/4 the sample image size sometimes causes crashes.

3. Quilting with block size of 1 does not invoke pixel-by-pixel resampling and may result in undefined behavior although it is unlikely cause a crash.

4. Make certain all ENVI files are: 1) band interleaved by pixel (bip) 2) of type unsigned int (12) 3) and unix byte order (1).

5. S/P applied to input and output textures of equal size sometimes fails.

6. Make sure the output file names are unique when using batch processing.

7. An invalid border around the synthetic texture will be observed when using default S/P method on color and multi-spectral textures. The reason

is that a resampling operation based on a template match is used to generate the residual spectra. This template is not allowed to wrap the edge of the synthetic image.

8. All textures must be square!

8 Source Tree

All the code provided with this distribution is compatible with the ANSI standard for C and C++ code. Although C++ is used to implement complex data structures, a majority of the core system is effectively C source code. In effect the only difference is that the conveniences of overloading and templates are used extensively in the code.

As mentioned, the source is organized into sub-projects. The top level *src* directory is the synthesis code used to implement the newTex framework. The additional code is 3rd party and is used to implement specific support functionality. In general this code is not to be modified and is used and provided as is. For completeness a description of each 3rd party library is provided here.

8.1 TIPS Library

The most significant portion of 3rd party code is the *Template Image Processing System* i.e. $TIPS$ library written by Thomas F. El-Maraghi. This library contains all the image and matrix data structures and support functionality used in the newTex system. The core data structures are $cImage < X >$, and $cMultiChannel < X >$ which are used to hold monochrome and multi-channel images respectively. Notice that these are templated C++ classes meaning their underlying data types are parameterized until compile time.

In addition to the core data structures this library provides a majority of the image processing software. Things like convolution, FFT and histogram matching are all provided by the $TIPS$ library. For those who are interested, the $TIPS$ library doesn't use efficient data structures for fast copy, access, and math operations on large matrices. It also contains no sparse matrix support.

8.2 ANN Library

The *ANN* [1] library provides an implementation for addressing the approximate nearest neighbor problem. Specifically the KD-tree implementation is used extensively in the code. This is a particularly efficient implementation that has the added benefit of being made *exact* if that level of precision is required.

8.3 IPI Library

A few files from CIS's *IPI* library are borrowed for inverting matrices are performing least-squares estimation.

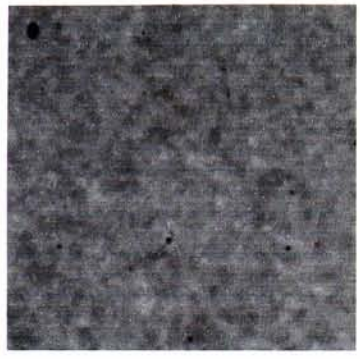
8.4 Documentation

Extensive code documentation is provided that was generated using the doxygen system. All the core code in the top level /src directory is fairly well documented included detailed descriptions of functions and there arguments as well as data and function dependencies. All documentation is in html and can be accessed using any browser by opening the top level html file in the /doc directory.

8.5 Modifying the Code

Users are encouraged to modify the source. Some easy steps include changing the top level function calls in *texture.cpp*. For instance try swapping a quilting variant for the monochrome S/P step when synthesizing color and multi-spectral texture. Another easy place to experiment is to add a call to *EnforceSpectralCovariance()* virtually anywhere in the synthesis process. Of particular interest is making this call with whitened vs. non-whitened multi-channel textures.

After making a few top level modifications like those just suggested, the user may want to look at the file *in_paint.h*. This file contains a lot of powerful functionality not all of which is directly implemented. This includes Wei and Levoy [6] synthesis and in-painting. There is also an implementation of the Heeger and Bergen [3] algorithm in *ttexture.h*.



(a)

Figure 2: Examples of bad choices for sample textures.

9 Obtaining New Data

While a number of good sources are available for monochrome and color textures, there is little in the way of multi-spectral texture. The multi-spectral textures on the distribution CD where all obtained from RIT's MISI airborne sensor. This data is an excellent source for textures.

When obtaining new textures make sure that the size is sufficient. Basically 64x64 is a minimum but anything bigger than 256x256 is probably unnecessary. The trick is to ensure that the prominent spatial features all repeat a number of times in the sample texture. The thing to avoid is textures that are not homogeneous meaning they contain large scale features that don't repeat in a predictable fashion. This isn't just a bad idea but violates the assumptions that the synthesis algorithms are based on. In some sense the succuss of the synthesis operation is dependant on the succuss of the sample texture selection. As an example consider the textures in Figure 2.

References

[1] ANN: Library for Approximate Nearest Neighbor Searching Version 0.2 (Beta release) http://www.cs.umd.edu/ mount/ANN/

[2] Brodatz, P. *A Photographic Album fo Artists and Designers.* Dover, New York, 1996.

[3] Heeger, D. J., AND Bergen, J. R. *Pyramid-based texture analysis/synthesis.* In Proceedings of SIGGRAPH 95 (Anaheim, California), August 6 11, 1995.

[4] Portilla, J. and Simoncelli, E.P. *A parametric texture model based on joint statistics of complex wavelet coefficients.* Int. J. Computer Vision, 40:49 71, 2000.

[5] VisTex *An online collection of visual textures.* MIT Media Laboratory, 1995. Available from http://www.white.media.mit.edu/vismod/VisionTexture/vistex.html

[6] Wei, Li-Yi and Levoy, Marc *Fast texture synthesis using tree-structured vector quantization.* In SIGGRAPH 2000, pages 479-488, 2000.